CLASSCLEANER: A QUANTITATIVE METHOD FOR VALIDATING PEPTIDE IDENTIFICATION IN LC-MS/MS WORKFLOWS

Melissa C. Key

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in the Department of Biostatistics,

Indiana University

May 2020

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

Benzion Boukai, Ph.D., Chair

January 17, 2020

Susanne Ragg, M.D., Ph.D.

Barry Katz, Ph.D.

Amber Mosley, Ph.D.

DEDICATION

To My Beloved Family.

ACKNOWLEDGMENTS

I need to start this by thanking my parents. For two years, they allowed me and my (then) two children into their home, helped watch them while I attended class, and provided meals while I stayed there. When my computer was dying, my father donated one to me. Their support has been steadfast and strong, and I could not have done any of this without them.

Dr. Susanne Ragg provided the financial support for most of my graduate career. Beyond that, she has been a mentor, an advisor, and a constant supporter of me throughout my graduate career. Without her, none of this would have been possible.

Dr. Ben Boukai, thank you for serving as my advisor. I still remember you telling me that there is a Ph.D inside me. When I've struggled, I remember your words, and I've used that to power through. Your direction and advice have been invaluable.

To the other members of my committee - thank you so much for your time.

To the Biostatistics and Mathematics departments at IUPUI - I didn't have the opportunity to spend much time getting to know everyone, but I enjoyed every minute of being there. Thank you for providing such a welcome community.

Finally, I'd like to thank Olga Vitek for introducing me to proteomics. It's proven to be an amazing field.

Last, but certainly not least, to my husband. This has been more years in the making than you ever would have guessed. Thank you for your patience, your support, and your help with keeping kids busy while I've worked.

Melissa C. Key

CLASSCLEANER: A QUANTITATIVE METHOD FOR VALIDATING PEPTIDE IDENTIFICATION IN LC-MS/MS WORKFLOWS

Because label-free liquid chromatography-tandem mass spectrometry (LC-MS/MS) shotgun proteomics infers the peptide sequence of each measurement, there is inherent uncertainty in the identity of each peptide and its originating protein. Removing misidentified peptides can improve the accuracy and power of downstream analyses when differences between proteins are of primary interest.

In this dissertation I present classCleaner, a novel algorithm designed to identify misidentified peptides from each protein using the available quantitative data. The algorithm is based on the idea that distances between peptides belonging to the same protein are stochastically smaller than those between peptides in different proteins. The method first determines a threshold based on the estimated distribution of these two groups of distances. This is used to create a decision rule for each peptide based on counting the number of within-protein distances smaller than the threshold.

Using simulated data, I show that classCleaner always reduces the proportion of misidentified peptides, with better results for larger proteins (by number of constituent peptides), smaller inherent misidentification rates, and larger sample sizes. ClassCleaner is also applied to a LC-MS/MS proteomics data set and the Congressional Voting Records data set from the UCI machine learning repository. The later is used to demonstrate that the algorithm is not specific to proteomics.

Benzion Boukai, Ph.D., Chair

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

A1AG1 alpha-1-acid glycoprotein 1.
A1AG1-A1AG2 alpha-1-acid glycoprotein 1 & 2.
A1AG2 alpha-1-acid glycoprotein 2.
AENN all-k edited nearest neighbors.
ANOVA analysis of variance.
APOA1 apolipoprotein A1.
APOB apolipoprotein B.
BBNR blame based noise reduction.
BH Benjamini-Hochberg.
BP-Quant Bayesian proteoform quantification.
CDF cumulative distribution function.
CERU ceruloplasmin.
CQA Congressional Quarterly Almanac.
CVCF cross-validated committees filter.
DCF dynamic classification filter.
EB edge boosting filter.
EF ensemble filter.
ENG editing with neighboring graphs.
ENN edited nearest neighbor.
EWF edge weight filter.
FDR false discovery rate.
FN false negative.
FOR false omission rate.
FP false positive.
GE generalized edition.
HARF high agreement random forest.
HC hierarchical clustering.
HEMO hemopexin.
HRF hybrid repair-remove filter.
I-C4.5 iterative C4.5 filter.
IBL instance-based learning.
INFFC iterative noise filter based on the fusion of classifiers.
IPF iterative partitioning filter.
LC liquid chromatography.
LC-MS/MS liquid chromatography-mass spectrometry.
M/Z mass-to-charge ratio.
MCMC Markov chain Monte Carlo.
MF mode filter.
MS mass spectrometry.
MSE mean squared error.
NN nearest neighbor.

ORBoost outlier removal boosting.
PDF probability density function.
PF partition filter.
PQPQ protein quantification by peptide quality control.
PRISM preprocessing instances that should be misclassfied.
PSM peptide-spectrum match.
PTM post-translational modification.
R-C4.5 robust C4.5 filter.
SVM support vector machine.
TN true negative.
TP true positive.
V-C4.5 voting C4.5 filter.

Chapter 1

Introduction

The research presented in this dissertation is motivated by an open problem in the quantification of proteins in a label-free shotgun proteomics work-flow. More generally, it presents a non-parametric solution to the problem of identifying instances that are outliers relative to the subset of instances assigned to the same class. This serves as a proxy for finding errors in the data set: instances for which the class label is recorded incorrectly, or where the measurements for a particular instance are sufficiently inaccurate as to render them uninformative. These errors can strongly influence downstream analyses which assume that all measurements are equally representative of the class (protein) being measured. To address this problem, I propose a procedure to find these outliers so that they may be removed or reassessed – improving the power and accuracy of the analyses of interest.

1.1 The motivating example: label-free shotgun proteomics

Quantitative label-free shotgun proteomics uses *liquid chromatography-mass spectrometry* (LC-MS/MS) to measure the relative abundance of many proteins simultaneously across a group of samples. Using LC-MS/MS, the relative abundance of many proteins from complex mixtures such as biological cells, tissues, and fluids (e.g. blood plasma) can be analyzed. As described by Aebersold and Mann (2003); Steen and Mann (2004); and Vitek (2009), the LC-MS/MS is extremely powerful in its ability to simultaneously measure many proteins. Conversely, proteins are not analyzed intact, which presents problems in determining which protein each measurement is associated with.

The first step in the LC-MS/MS workflow is to enzymatically cleave each protein into many pieces, known as peptides. What was already a complex mixture of proteins becomes an even more complex mixture of peptides, necessitating one or more steps separating this mixture into simpler mixtures. The last method of separation, known as *liquid chromatography* (LC), is coupled to the mass spectrometer. It consists of a small tube, called a column, packed with a material through which peptides from a single sample pass at a rate based on their hydrophobicity. The time at which a particular peptide passes through is thus predictable, allowing data from multiple samples to be aligned algorithmically. After exiting the column, peptides enter the mass spectrometer where they are ionized and the ion signal over a range of *mass-to-charge ratios* (M/Zs) is recorded.

The recorded ion signals are insufficient to map to a protein or (more specifically) a unique peptide sequence originating from a particular protein. To obtain this match, a second type of *mass spectrometry* (MS) scan is utilized. In this so-called MS2 scan (in contrast to the MS1 scans described above), a single peptide species is isolated based on its M/Z and bombarded until it splits into two pieces along its backbone. The fragments are analyzed by the mass spectrometer as before to obtain a unique "fingerprint" for the peptide. Thus, after each MS1 scan, MS2 scans are typically performed on the 10 highest peaks so long as they meet a minimum intensity threshold and haven't been previously analyzed. The raw output from each run is thus a collection of MS1 and MS2 scans over time.

Identification algorithms match the MS2 spectra to peptide sequences and their originating protein, allowing both the peptide sequence and protein to be identified. For sequenced genomes, *peptide-spectrum matches* (PSMs) are usually created by comparing the observed MS2 spectra to a database of theoretical spectra created using cDNA libraries, as described in Aebersold and Mann (2003); Steen and Mann (2004). The general idea of these algorithms can be summarized as follows:

1. Select a subset of peptides from the database matching the observed sequence on some initial criterion - often mass or the sequence of the first 3 to 4 amino acids.

2. Compare the experimental spectrum to the theoretical spectra from each peptide in the subset, and generate a numerical score for each match.

3. Accept the top scoring PSM as the correct identification.

4. Filter out less confident PSMs to control the *false discovery rate* (FDR). See Elias and Gygi (2007, 2010) for how this is done using decoy databases.

Using protein databases in the identification procedure is easier than de-novo methods and ensure that every identified peptide is associated with a protein. Conversely, their use has two important limitations. Correct identification is only possible if the true peptide sequence exists in the database. This presents a fundamental problem: including all possible variations of every single peptide has a multiplicative effect on the size of the database. Even if such a database could be stored and searched through in a reasonable amount of time, this multiplicatively increases the proportion of incorrect sequences to which each spectrum is compared, increasing the probability that the best PSM comes from an incorrect match. Balancing database completeness with database size is an open problem, but is outside the scope of this dissertation.

The second problem with protein databases is more subtle. Mass spectrometers are able to distinguish differences in peptides caused by alterations in the DNA sequence, RNA processing, and post-translational processing of the protein. When these different proteoforms (Smith and Kelleher, 2013) have dissimilar abundance patterns over the biological samples, the mass spectrometer can find peptides from each proteoform, but current algorithms group all patterns together into a single protein. To highlight the difficulty of this problem, consider a single protein with two proteoforms. Peptides from this protein may have one of three different abundance

patterns: one from peptides existing only in the first proteoform, one from peptides existing only in the second proteoform, and one from peptides shared by both. The most common abundance pattern may not be the one of greatest research interest. Handling within-protein heterogeneity across peptides is another open problem within proteomics that is only beginning to be addressed.

Labeling errors, in which a spectrum is incorrectly matched to a peptide sequence from one protein, when it actually originates from a peptide sequence from another protein, are thus endemic to label-free shotgun proteomics. Intensity measurements are also subject to error, especially when peptides are either highly abundant (saturated) or barely detected by the mass spectrometer. The goal of this research is thus to develop a method that can find and remove these peptides. Using cleaned data, I hope to improve the ability of other algorithms to group peptides into homogeneous clusters, thus strengthening our ability to cluster and classify samples, perform tests of differential abundance, and understand protein biology.

1.2 Existing approaches

Handling misidentified peptides has been approached using multiple strategies developed within the field of proteomics, and others can be easily adapted from other fields. Proteomic strategies have traditionally focused on robust methods of summarizing the available data over the peptides (Silva et al., 2006b,a; Polpitiya et al., 2008; Suomi et al., 2015), but alternatives including explicitly modeling identification errors (Lucas et al., 2012) and filtering (Forshed et al., 2011; Webb-Robertson et al., 2014) have also been proposed. Of these approaches, filtering has the advantage of producing a cleaned data set that can be used for all subsequent analyses, including differential abundance, and classification and clustering across samples. Filtering algorithms have also been designed to clean the training sets used in supervised machine learning, aka classification. These so-called classification filters perform a very

similar task to what is desired for proteomics data, but are typically evaluated by the performance of the resulting classifier, not on the accuracy of the cleaned training set.

1.2.1 Current proteomic strategies

In the proteomics community, peptide heterogeneity is typically studied in the context of differential abundance analyses. More specifically, the question of interest is how to summarize the data from many peptides into a single test statistic that accurately reflects the behavior of the protein. While some more recent work has allowed for more than one summary per protein, interpreting these results remains challenging.

If all peptides accurately reflect the intensity of a single protein with a single proteoform, the most efficient way to incorporate all data is a least squares or mixed effects analysis, as described by Clough et al. (2009). In the presence of noisy peptide data, this type of analysis causes the true signal to be averaged with signals from other sources, potentially leading to inaccurate, biased, or less powerful conclusions regarding the behavior of the protein. Averaging across multiple proteoforms can lead to results which do not reflect the truth of any proteoform.

Non-filtering methods

While filtering methods are the primary focus of this dissertation, two categories of non-filtering methods exist, namely robust and modeling methods. Robust methods summarize each protein by a small number of peptides with a high likelihood of accurately reflecting the abundance of the protein. The "Top-3" method (Silva et al., 2006a,b) uses the three peptides with the highest average intensity to represent each protein. Suomi et al. (2015) calculates a test statistic for each peptide, and uses the median test statistic to represent the protein. Polpitiya et al. (2008) presents software that can "roll up" peptide intensities into a protein average, which can potentially use

a larger proportion of the peptides. These methods involve scaling the peptide intensities (using a reference peptide or the median intensity across all samples) to create more uniform intensities, removing outliers based on Grubb's outlier test (Grubbs, 1969), and averaging the remaining peptides. Alternatively, they propose removing all peptides with intensities below a user-selected cut-off and averaging the remaining peptides. Conversely, by "rolling up" the peptide data into a single summary prior to differential abundance, all variation inherent in the peptide measurements is still ignored. Because these methods only use a small number of peptides, their power is very low relative to strategies using more proteins. Furthermore, they ignore proteoforms, with only a single summary obtained per protein.

To explicitly model proteoforms and misidentification into differential abundance model, Lucas et al. (2012) first proposes a latent variable model to predict peptide intensity using a mixture of sample-specific factors and protein identity. Using a Bayesian framework, the authors use a Dirichlet distribution to put a prior on the protein identity of each peptide. The observed protein identity of each peptide is accounted for by setting the corresponding parameter of the Dirichlet prior to 100 - 1000 times the value given to other proteins. The model is fit using *Markov chain Monte Carlo* (MCMC). Henao et al. (2013) extends this idea to incorporate correlation across proteins, as exists when proteins are involved in the same biological pathways. Modeling methods such as this have high potential, but typically are computationally expensive.

Filtering Strategies

At present, there are two proteomic-specific algorithms designed to group peptides into proteoforms while simultaneously removing outliers. For each protein, *protein quantification by peptide quality control* (PQPQ) (Forshed et al., 2011) starts by defining "high confidence peptides" based on the reported confidence scores provided

by identification software and their correlation with other high confidence peptides. These peptides are clustered using hierarchical clustering and correlation distance. The peptide with the highest intensity is selected as the representative peptide of the cluster. All peptides in the protein are then compared to each representative peptide, and those which are sufficiently close are added to the corresponding cluster. Clusters are then consolidated to reduce overlap.

The *Bayesian proteoform quantification* (BP-Quant) procedure (Webb-Robertson et al., 2014) takes as input the results of a set of significance tests, coded as $(-1, 0, 1)$, where -1 corresponds to a significant negative result, 1 is a significant positive result, and 0 is a non-significant result. The algorithm then groups the peptides into potential proteoforms based on their significance pattern, and uses Bayesian modeling to determine which proteoforms are actually present in the protein. Their approach focuses attention on the comparisons of interest, groups peptides according to the significance results, and simultaneously removing peptides showing unusual signals.

While Webb-Robertson et al. (2014) includes a comparison between PQPQ and BP-Quant, this is on a dilution series with no natural biological variation. performance of these algorithms on real data has not been examined to date.

1.2.2 Classification filtering algorithms

Classification filtering algorithms are an alternative source of potentially promising algorithms which could be applied to proteomics data. These algorithms have been developed to address a decline in performance of classification algorithms in the presence of labeling errors, that is, when an instance is assigned an incorrect label (Quinlan, 1986; Zhu and Wu, 2004; Sáez et al., 2014). Frénay and Verleysen (2014) claims that label noise in the training set has been found to decrease the accuracy of the resulting classifier, increase the complexity of the inferred models, and increase the number of training samples required. Filtering, by identifying and removing misla-

beled instances, is one of three broad groups of solutions to this problem (Frénay and Verleysen, 2014).

For easier discussion, the classification filtering algorithms are grouped into five broad categories. Boosting algorithms are based on the adaBoost classification algorithm. *Instance-based learning* (IBL) algorithms are based designed for instance based learning, most commonly *nearest neighbor* (NN). Voting methods combine information obtained using multiple algorithms or repeating the same algorithm on different subsets of the data. Iterative solutions repeatedly apply one or more algorithms until some convergence criterion has been satisfied. The last category consists of algorithms which do not fit into any of the above groups.

Boosting Methods

AdaBoost (Freund and Schapire, 1997) is a binary classification algorithm that works by iteratively generating a series of weak classifiers that together creates a strong classifier. After each iteration, the samples and classifiers are weighted according to their accuracy, with accurate classifiers and misclassified samples receiving larger weights. The final classifier is created by taking the weighted sum of all classifiers in the model.

Schapire (1997) extends the original adaBoost classifier to multi-class problems using output codes. Briefly, for each iteration, the labels are mapped to $\{0, 1\}$ using a random process. The weak learners are trained on the examples using the binary labels as in the original algorithm, with the classifier weights modified to account for the utility of the labeling mechanism.

The idea behind filtering methods based on adaBoost is that instance with consistently large weights are likely to be mislabeled. *Outlier removal boosting* (ORBoost) (Karmaker and Kwek, 2005) removes instances whose weights are larger than a specified threshold after each iteration. *Edge boosting filter* (EB) (Wheway, 2001) weights

the incorrect predictions of all instances by the respective classifier weights. If the total of these meets a specified threshold and/or is in the top $q\,\%$ of all so-called edges, the instance is considered misclassified.

While the original adaBoost algorithm is designed for binary classification, both papers address multi-class problems. EB uses a pruned C4.5 decision tree (Quinlan, 1993) instead of a weak classifier to extend the algorithm to multi-class problems. The procedure stops if any classifier misclassifies over half of the data set. Karmaker and Kwek (2005) recommends incorporating output codes to extend ORBoost to the multi-class case, but do not test this in practice.

The biggest concern with boosting methods is their ability to cope with many hundreds proteins with widely disparate sizes. Each weak classifier must classify at least half of the peptides correctly in order for the algorithms to be valid. If peptides misclassified by these algorithms are disproportionately from smaller proteins, the algorithm will tend to remove smaller proteins from the data set.

Instance based learning algorithms

IBL algorithms perform classification by comparing new instances to a set of instances in the training set (Aha and Kibler, 1989). The classic example is the k-NN algorithm (Cover and Hart, 1967). Reducing the size and complexity of the training set by removing mislabeled instances can improve both the speed and quality of the resulting classification algorithm. Conversely, removing too many instances can degrade the performance of the algorithm.

Wilson (1972) proposed *edited nearest neighbor* (ENN) as a method of reducing the size of the training set 1-NN classification. Using ENN, each instance in the training set is compared to the remaining points in the training set using k-NN, where $k > 1$. Instances are removed from the comparison set if the given class is different from the class predicted by the majority of its k nearest neighbors. While

primarily intended as a method of reducing the size of the comparison set, ENN also has the effect of removing misclassified instances. Koplowitz and Brown (1981), concerned that ENN removes too many samples, proposed *generalized edition* (GE) as a modification of the original algorithm. Using GE, the class of each instance is set to the class observed in at least $k' \geq \frac{K+1}{2}$ of the nearest neighbors and removed otherwise. Conversely, the *all-k edited nearest neighbors* (AENN) algorithm extends ENN by applying it for $k = 1, \ldots, k_{\max}$. Only instances retained by all filters are retained in the final algorithm.

Blame based noise reduction (BBNR), (Delany and Cunningham, 2004; Pasquier et al., 2005) was motivated by spam filters. They found that training using spam emails that are too similar to legitimate emails could cause misclassifications even though the emails were correctly identified as spam. To address this, BBNR tracks the effect each instance has as a nearest neighbor to other points. Their algorithm penalizes instances that contribute to other instances becoming misclassified, while rewarding observations that contribute to instances getting classified correctly. BBNR removes instances for that the penalty is larger than the reward – those instances that do more harm than good.

Editing with neighboring graphs (ENG) (Sánchez et al., 1997), the final IBL algorithm considered, uses proximity graphs instead of NN to determine that instances are used to judge class accuracy. Instead of a fixed number of nearest neighbors, ENG assigns edges between points if no other points come between them. (How this is defined is dependent upon the type of graph selected, and the reader is directed to the paper, Sánchez et al. (1997), to learn more about these methods.) The neighborhood around an instance consists of the set of instances connected by an edge. Instances are removed if the majority of instances in their neighborhood classify them incorrectly.

IBL algorithms generally have dual goals of simplifying and reducing the size of the training set. Consequently, while they can be used for proteomics filtering, they may greatly reduce the number of peptides per protein. This can lead to a lower power than possible with less aggressive filters.

Voting methods

Voting methods, first proposed by Brodley and Friedl (1996b), generate predicted classes multiple times under different conditions, with each predicted class counting as one "vote". Using a majority-vote filter, instances are removed if they are misclassified in at least half the votes. A consensus filter removes instances only if they are misclassifed by every vote. The majority and consensus filters can be used to filter votes generated by any of the voting methods described below.

Brodley and Friedl (1996b) proposed the *ensemble filter* (EF), which generates votes using three different classifiers (a decision tree, k-NN, and linear machine) and uses four-fold cross validation to determine whether instances are misclassified. More generally, the authors suggest that any set of classifiers can be use, but they recommend selecting algorithms use different classification strategies to maximize the independence of each method. As stated by the authors, their goal is to find instances that are outliers in *any* model.

Many variations of this general procedure have been developed. In the *hybrid repair-remove filter* (HRF) algorithm, Miranda et al. (2009) uses an ensemble of a *support vector machine* (SVM), a k-NN, an artificial neural network, and a decision tree. In the *dynamic classification filter* (DCF), Garcia et al. (2012) selects the three classifiers with the most similar results out of seven options (SVM, k-NN, CART, C4.5, random forest, naïve Bayes, and multilayer perceptron). The authors explain their criteria for picking filters as

> We then choose the m models with more similar predictions on the training data to compose the ensemble, that is, those classifiers that agree most in their predictions. We expect that these will be the classifiers with better joint ability to identify class mislabelings.

Both HRF and DCF allow the labels of consistently misclassified instances to be changed instead of removed, so as to retain the original size of the data set.

A different variation of the voting method generates votes using different subsets of the training set. This is similar to random forest classification, which works by randomly generating many decision trees using varying subsets of the training set. New instances are classified using all these decision trees, and the class labeled is obtained by taking the label made by the majority of trees. *High agreement random forest* (HARF) (Sluban et al., 2010) uses this same principle for filtering, but filters out instances given the wrong label in 60 % to 90 % of the trees. Verbaeten (2002) and Verbaeten and Assche (2003) splits the data into k cross-validation subsets, and trains a decision tree[1] on the sets created by leaving out one subset. In the *voting C4.5 filter* (V-C4.5) (Verbaeten, 2002), the class of each instance is predicted based on the $k-1$ trees that included the instance in the training set. In the *cross-validated committees filter* (CVCF) (Verbaeten and Assche, 2003), all k trees are used.

For large distributed data sets, many of these methods are difficult to implement because the entire data set is not available. The *partition filter* (PF) algorithm (Zhu et al., 2003) addresses this by training a decision tree on each partition of the data separately[2]. For an instance to be marked misidentified, it must be mis-classified by the locally trained decision tree *and* the majority or consensus using the other filters. The PF algorithm can also be applied to non-distributed data sets by splitting it

[1] In the paper, Verbaeten (2002) uses the Tilde algorithm, which extends the C4.5 algorithm (Quinlan, 1993) to inductive logic programming. Specifically, the tests in each node are logical queries instead of attribute-value tests.

[2] In generating these decision trees, the authors stress the importance of developing "good rules" – by which they mean rules that are extendable to other partitions. In particular, rules that only apply to a small subset of instances and those that do not discriminate sufficiently well are not "good rules".

into several subsets at random and training each algorithm on each subset separately. For proteomics data, there is a question on whether each subset is sufficiently large to accurately train the data set, especially for smaller proteins which may not be represented in all subsets.

In the original conception of voting methods from Brodley and Friedl (1996b), the importance of independent votes is stressed. Most other implementations of this method fail to achieve this in practice. DCF goes as far as to select classifiers producing similar results. Classification trees based on the same data set are necessarily correlated, so HARF, CVCF, and V-C4.5 all have some correlation inherent in them. The problem with this dependence is that similar methods are likely to produce the same list of outliers, because they make similar assumptions about the instance. Likewise, when there is a large amount over overlap between the input data sets, the resulting models will likely be similar, and produce similar lists of misclassified instances. These concerns are relevant regardless of the application, and are not specific to a proteomics.

Iterative methods

Iterative methods, as the name suggests, repeatedly call a filtering algorithm (such as those listed above). They use the filtered data from the previous iteration as an input in the next iteration until some specified criteria is met. The *robust C4.5 filter* (R-C4.5) method (John, 1995) removes all misclassified instances from a pruned decision tree, and regenerates the tree repeatedly until all instances are predicted correctly. *Iterative C4.5 filter* (I-C4.5) (Verbaeten, 2002) extends this idea by replacing a single decision tree with the V-C4.5 method described above. The *iterative partitioning filter* (IPF) repeats the PF until a given stopping criterion is reached (Khoshgoftaar and Rebours, 2004). The *iterative noise filter based on the fusion of classifiers* (INFFC) performs filtering twice in each iteration, with a variant of the EF algorithm applied at

each stage (Sáez et al., 2014). The first application of the filter is used to remove the noisiest instances before performing second "noise-free" filtering is performed. Results from the second round of filtering are input into a score function that discriminates between clean and noisy instances. The cleaned results are used as input for the next round of filtering.

Because iterative methods provide more opportunities to remove instances, these procedures would be expected to produce a greater reduction in both correctly labeled and incorrectly labeled peptides. Furthermore, they are also time-intensive because they require repeatedly rebuilding decision trees.

Other

Other algorithms have been developed which do not fit in one of the above categories. *Preprocessing instances that should be misclassfied* (PRISM) Smith and Martinez (2011) uses four heuristics based on a combination of nearest neighbors, C4.5 decision trees, and class probabilities to assess whether or not to remove each instance.

The *edge weight filter* (EWF) Muhlenbach et al. (2004) represents the data as a proximity graph. For each instance, connected edges are weighted based on distance and a score generated by summing up the edge weights from edges connecting to instances from other classes. When the probability of obtaining the observed score from a correctly labeled instance is sufficiently low, the instance is filtered out.

Du and Urahama (2009, 2011) proposed the *mode filter* (MF) as a method of filtering out noise in image analysis problems, although the method can be extended to general use. For each instance, the algorithm seeks to find the class label that maximizes a function that incorporates the class labels of other instances in a local neighborhood and their similarity (distance) to the instance of interest.

1.2.3 Implementation

The large number of proteins and the extreme size discrepancy in the number of peptides belonging to each protein make the computational aspects of any proposed filtering algorithms an important characteristic to consider in evaluating these algorithms. Inefficient or poorly written algorithms can result in increased memory usage or longer running times on small data sets. With respect to proteomics data, these algorithms will often crash due to memory errors, or require run times on the order of days or longer.

All of the classification filtering algorithms describe here are available in R (R Core Team, 2017) using the NoiseFiltersR package (Morales et al., 2016), which is available on CRAN. An R implementation of BP-Quant was available for download at https://github.com/PNNL-Comp-Mass-Spec/BP-Quant. Most of these algorithms were modified to decrease their run time, although inefficiencies remain. An implementation of the PQPQ algorithm R was adapted from the published Matlab code in order to facilitate comparisons across all comparative algorithms. Table C lists all algorithms discussed in this dissertation.

1.3 Scope

The purpose of this dissertation is to develop an algorithm that can identify peptides that are likely to be either mis-identified or poorly quantified based on the quantitative measures of each peptide across a number of samples. In particular, an algorithm is proposed that filters out poorly-characterized instances from each group (e.g. proteins) in a system where the number of groups is large. The proposed algorithm is then compared to alternative filtering methods using simulations and data from a proteomics study on Sickle Cell Anemia.

1.4 Outline of remaining chapters

The outline of the remaining chapters is as follows. In Chapter 2, I introduce classCleaner, a novel solution to detecting mislabeled instances and provide a theoretical justification for its validity. Chapter 3 uses simulated data to assess the performance of classCleaner and compares its results to those of other available methods. In Chapter 4, it is demonstrated how classCleaner can be used to clean LC-MS/MS data, using a data set comprised of measurements from 8219 peptides and 120 samples in a study on sickle cell anemia. To exhibit how classCleaner can be applied to a non-LC-MS/MS data set, Chapter 5 contains an analysis of the Congressional Voting Records data set from the UCI Machine Learning Repository (Dheeru and Karra Taniskidou, 2017). Finally, Chapter 6 concludes with a discussion of the results.

Chapter 2

A filtering method based on the binomial distribution

Consider a data set on N peptides, of which $N_1, N_2, \ldots, N_{K-1}$ are presumed to belong to proteins $P_1, P_2, \ldots, P_{K-1}$ respectively, with $N_k > 1$ for $k = 1, \ldots, K-1$. Let P_K be a 'mega-protein' consisting of the N_K peptides unassigned to a specific protein or peptides which are the sole representative of a protein, such that $N_K = N - \sum_{k=1}^{K-1} N_k$. For simplicity, the shorthand notation $i \in P_k$ indicates that peptide i belongs to protein P_k while $i \notin P_k$ indicates that peptide i does not belong to protein P_k. Let $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})'$ be the (vector) of observed intensities from peptide i, (i = 1,...,N), across n (independent) samples. The available data is thus $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ is an $n \times N$ matrix of such observed intensities. The 'distance' between any two peptides with observed intensities $\mathbf{x}_i$ and $\mathbf{x}_j$ can thus be measured using any standard distance or quasi-distance function,

$$d_{ij} = \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \geq 0. \tag{2.1}$$

For instance, d_{ij} could be the quasi-distance defined by the correlation between $\mathbf{x}_i$ and $\mathbf{x}_j$, $d_{ij} = 1 - r_{ij}$, where

$$r_{ij} := \text{cor}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{\ell=1}^{n} (x_{i\ell} - \bar{x}_{i\cdot})(x_{j\ell} - \bar{x}_{j\cdot})}{\sqrt{\sum_{\ell=1}^{n} (x_{i\ell} - \bar{x}_{i\cdot})^2 \sum_{\ell=1}^{n} (x_{j\ell} - \bar{x}_{j\cdot})^2}}. \tag{2.2}$$

Here $\bar{x}_{i\cdot} = \sum_{\ell=1}^{n} x_{i\ell}/n$, for each $i = 1, \ldots, N$.

Let $\mathbf{D} = \{d_{ij} : i, j = 1, \ldots N\}$ be the $N \times N$ (symmetric) matrix comprised of these between-peptide observed distances. Without loss of generality, assume that the entries of $\mathbf{D}$ are ordered such that the first N_1 entries belong to P_1, the next N_2

entries belong to P_2, etc. Accordingly, partition $\mathbf{D}$ as

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{D}_{12} & \dots & \mathbf{D}_{1K} \\ \mathbf{D}_{21} & \mathbf{D}_{22} & \dots & \mathbf{D}_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{D}_{K1} & \mathbf{D}_{K2} & \dots & \mathbf{D}_{KK} \end{bmatrix}, \tag{2.3}$$

where $\mathbf{D}_{kk}$ is the $N_k \times N_k$ matrix of between-peptide distances *within* protein P_k and the elements of $\mathbf{D}_{k_1k_2}$ represent the distances between the N_{k_1} peptides belonging to P_{k_1} and the N_{k_2} peptides belonging to P_{k_2}. Note in particular that $\mathbf{D}_{k_1k_2} \equiv \mathbf{D}'_{k_2k_1}$.

To begin with, consider at first protein P_1 and the N_1 peptides initially assigned to it. For a fixed i, $i = 1, \dots, N_1$, let $i \in P_1$ be a given peptide in protein P_1 and set $\mathbf{d}_i := (\mathbf{d}_i^{(1)}, \mathbf{d}_i^{(2)}, \dots, \mathbf{d}_i^{(K)})$, be the i^{th} row of $\mathbf{D}$, where $\mathbf{d}_i^{(1)} := (d_{i1}, \dots, d_{iN_1})'$ and for $k \geq 2$, $\mathbf{d}_i^{(k)} := (d_{i(1+\sum_{j=1}^{k-1} N_j)}, \dots, d_{i(\sum_{j=1}^{k} N_j)})'$. Clearly, $\mathbf{d}_i^{(k)}$ is the i^{th} row of $\mathbf{D}_{1k}$ for $k = 1, \dots, K$. Next, for the protein P_1, consider the stochastic modeling of the elements $\mathbf{D}_{11}, \mathbf{D}_{12}, \dots, \mathbf{D}_{1K}$. For each peptide, $i \in P_1$, the observed within-protein distances from it to the other $N_1 - 1$ peptides in P_1 are assumed to be *i.i.d.* random variable according to a protein-specific distribution, $G(\cdot)$, so that

$$\big(d_{i1}, \dots, d_{i(i-1)}, d_{i(i+1)}, \dots, d_{iN_1}\big) \underset{i.i.d.}{\sim} G_i(\cdot), \tag{2.4}$$

(since $d_{ii} \equiv 0$). Here, the *cumulative distribution functions* (CDFs) $G_i(\cdot)$ are defined for each fixed $i \in P_1$, and any $j \in P_1$ as

$$G_i(t) \equiv G(t) := \Pr(d_{ij} \leq t \,|\, i \in P_1,\ j \in P_1,\ j \neq i), \quad \forall t \in \mathbb{R}. \tag{2.5}$$

Our notation in (2.5) stresses that the distribution of distances from each individual peptide to the remaining peptides in P_1 are assumed to be identical. Similarly, the distances between the given peptide, $i \in P_1$, and the N_k peptides in protein P_k, $k = 2, \dots, K$, are also *i.i.d.* random variables according to some distribution $F^{(k)}(\cdot)$, so

that

$$(d_{ij}) \underset{i.i.d.}{\sim} F^{(k)}(\cdot), \quad \forall j \in P_k, \quad (j = 1 \ldots, N_k). \tag{2.6}$$

Here $F^{(2)}, F^{(3)}, \ldots, F^{(K)}$ are $K-1$ distinct CDFs defined for each $i \in P_1$, and any $j \in P_k$ as

$$F^{(k)}(t) = \Pr\left(d_{ij} \leq t | \, i \in P_1, \ j \in P_k\right), \quad \forall t \in \mathbb{R}. \tag{2.7}$$

Throughout this work, $G(\cdot), F^{(2)}(\cdot), \ldots, F^{(K)}(\cdot)$ are assumed to be continuous distributions with *probability density functions* (PDFs) $g(\cdot), f^{(2)}(\cdot), \ldots, f^{(K)}(\cdot)$ respectively. If we further assume that all the N_1 peptides from protein P_1 are equally representative of the true intensity across all n samples, we would expect that the distances between any two peptides from within protein P_1 are *stochastically smaller* than distances between peptides from within P_1 and peptides associated with protein P_k where $k \neq 1$. Accordingly,

Assumption 1. *For each* $k, \quad k = 2, \ldots, K,$

$$\Pr(d_{ij} \leq t | \, i \in P_1, \ j \in P_k) \leq \Pr(d_{ij} \leq t \, | \, i \in P_1, \ j \in P_1, \ j \neq i) \tag{2.8}$$

or equivalently,

$$F^{(k)}(t) \leq G(t) \quad \forall t \in \mathbb{R}\,.$$

In light of (2.7), the distribution of distances from the i^{th} peptide in P_1 to any other random peptide J, selected uniformly from among the $N - N_1$ peptides not in P_1, is thus the mixture,

$$\bar{F}(t) := \Pr\left(d_{iJ} \leq t | \, i \in P_1, J \notin P_1\right) = \frac{1}{N - N_1} \sum_{k=2}^{K} N_k F^{(k)}(t), \tag{2.9}$$

where, by assumption, $\Pr(J \in P_k | \, J \notin P_1) := N_k/(N - N_1)$. Further, if I is a randomly selected peptide in protein P_1, selected with probability $\Pr(I = i | I \in P_1) =$

$1/N_1$, for $i = 1, 2, \ldots, N_1$, then it follows that

$$\begin{aligned}\Pr(d_{IJ} \leq t | I \in P_1,\ J \notin P_1) &= \sum_{i=1}^{N_1} \frac{1}{N_1} \Pr(d_{iJ} \leq t | i \in P_1,\ J \notin P_1) \\ &= \sum_{i=1}^{N_1} \frac{1}{N_1} \bar{F}(t) \\ &\equiv \bar{F}(t) \end{aligned} \tag{2.10}$$

Similarly, if I and J represent two distinct peptides, both randomly selected from P_1 with $\Pr(I = i,\ J = j | I \in P_1,\ J \in P_1) = 1/N_1(N_1 - 1)$, then

$$\begin{aligned}\bar{G}(t) &:= \Pr(d_{IJ} \leq t | I \in P_1,\ J \in P_1,\ J \neq I) \\ &= \sum_{i=1}^{N_1} \sum_{j=1, j\neq i}^{N_1} \Pr(d_{ij} \leq t | i \in P_1,\ j \in P_1,\ j \neq i) \\ &= \sum_{i=1}^{N_1} \sum_{j=1, j\neq i}^{N_1} \frac{1}{N_1(N_1 - 1)} G_i(t) \\ &\equiv G(t). \end{aligned} \tag{2.11}$$

It follows by Assumption 1 that

$$\bar{F}(t) \leq \bar{G}(t),\ \forall t \in \mathbb{R}. \tag{2.12}$$

Now, for any $t \in \mathbb{R}$ define

$$\psi(t) := \bar{G}^{-1}(1 - \bar{F}(t)). \tag{2.13}$$

As will be shown in Lemma 1, since $\bar{G}$ and $\bar{F}$ are continuous there is a unique solution, t^*, such that $t^* = \psi(t^*)$ and

$$\bar{G}(t^*) = 1 - \bar{F}(t^*) := \tau. \tag{2.14}$$

For $\bar{F}(t)$ and $\bar{G}(t)$ to be differentiable, (2.14) assumes that $\bar{G}(t^*) = \tau > 0.5$. The lemma is stated as follows, with the proof given in Appendix A.

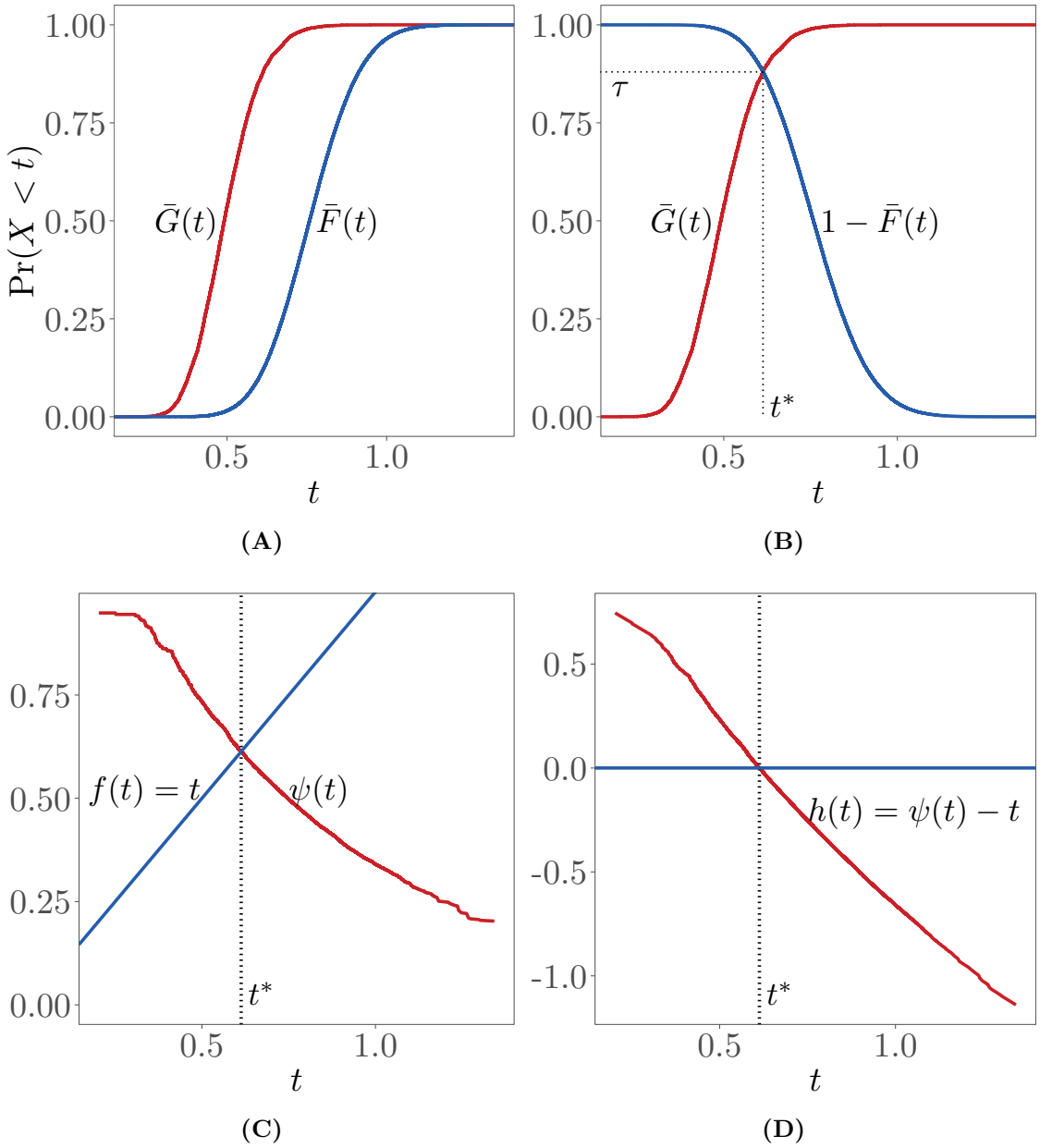


Figure 2.1: For continuous CDF functions $\bar{F}$ and $\bar{G}$ (A), the functions $\bar{G}(t)$ and $1-\bar{F}(t)$ will cross exactly once (B). When $\bar{G}(t) \geq \bar{F}(t)$ for all $t \in \mathbb{R}$, then this point, t^*, has the property that $\tau := \bar{G}(t^*) = 1-\bar{F}(t^*) > 0.5$. At t^*, $\psi(t^*) = \bar{G}^{-1}(1-F(t^*)) = t^*$ (C) and thus $h(t^*) = \psi(t^*) - t^* = 0$ (D).

Lemma 1. *Define* $\psi(t) = \bar{G}^{-1}(1 - \bar{F}(t))$ *as in (2.13), and set* $h(t) = \psi(t) - t$. *Then* $h(t)$ *has a unique root at* t^* *at which*

$$\psi(t^*) = t^*.$$

Figure 2.1 illustrates the relationship between $\bar{G}(t)$, $\bar{F}(t)$, $1 - \bar{F}(t)$, and $\psi(t)$. As will be shown below, the value of t^* serves as a cut-off point to differentiate between the distribution governing distances between peptides in protein P_1 and the distribution of distances going from peptides in P_1 to all remaining peptides.

2.1 The Filtering Procedure

2.1.1 Constructing the test

Consider at first protein P_1 and the N_1 peptides initially assigned to it. Based on the available data, we are interested in constructing a testing procedure for determining whether or not a given peptide that was assigned to protein P_1 should be retained or be removed from it (and potentially be reassigned to a different protein). That is, for each selected peptide from the list $i \in \{1, 2, \ldots, N_1\}$ of pre-labeled (or proposed) peptides in P_1, consider the statistical test of the hypothesis

$$\mathcal{H}_0^{(i)} : i \in P_1 \quad \text{(the initial identification is correct)}$$

against

$$\mathcal{H}_1^{(i)} : i \notin P_1 \text{ (the initial identification is incorrect)},$$

for $i = 1, \ldots, N_1$. The final result of these successive N_1 hypotheses tests, is the set of all those peptides in P_1 for which $\mathcal{H}_0^{(i)} : \ i \in P_1$ was rejected and thus, providing the set of those peptides in P_1 which were deemed to have been misidentified. As will be shown below, the proposed successive testing procedure is constructed so as

to control the maximal probability of a type I error, while minimizing the probability of a type II error.

Towards that end, define

$$Z_i \equiv \sum_{j \in P_1,\ j \neq i} \mathrm{I}[d_{ij} \leq t^*] = \sum_{j=1,\ j \neq i}^{N_1} \mathrm{I}[d_{ij} \leq t^*] \tag{2.15}$$

for each $i \in \{1, 2, \ldots, N_1\}$ where t^* is defined by (2.14) and I is the indicator function. In light of the relation (2.12), Z_i will serve as a test statistic for the above hypotheses. The distribution of Z_i under both the null and alternative hypotheses can be explicitly defined as Binomial random variables, as shown in the following pair of lemmas (proofs are provided in Appendix A).

Lemma 2. *If $i \in P_1$, then Z_i is a sum of i.i.d, r.v.s, and for each $i = 1, 2, \ldots, N_1$,*

$$Z_i|_{\mathcal{H}_0^{(i)}} \sim Bin\left(N_1 - 1, \bar{G}(t^*)\right) \equiv Bin(N_1 - 1, \tau).$$

such that $\mathrm{E}\left[Z_i \,\middle|\, \mathcal{H}_0^{(i)}\right] = (N_1 - 1)\bar{G}(t^*)$.

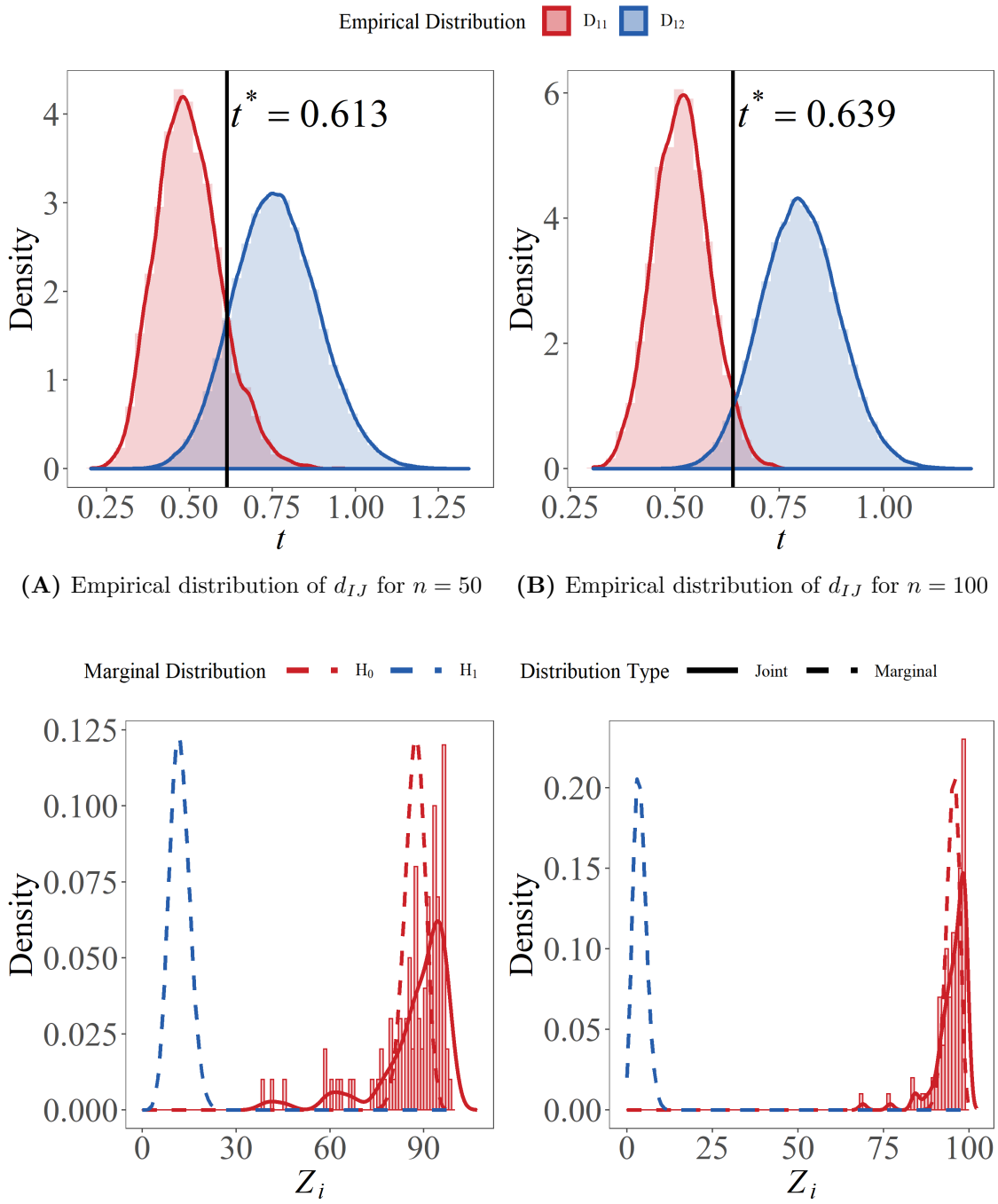
Lemma 3. *If $i \notin P_1$, but $j \in P_1$ for $j = 1, \ldots, i-1, i+1, \ldots, N_1$, then Z_i is a sum of independent r.v.s with distribution*

$$Z_i|_{\mathcal{H}_1^{(i)}} \sim Bin(\left(N_1 - 1, \bar{F}(t^*)\right) \equiv Bin(N_1 - 1, 1 - \tau)$$

where $\mathrm{E}\left[Z_i \,\middle|\, \mathcal{H}_1^{(i)}\right] = (N_1 - 1)\bar{F}(t^*)$.

Using simulated data (complete details on the data generation procedure are given in Chapter 3), Figure 2.2 shows histograms of the observed Z_i and the corresponding marginal distribution of each Z_i under both hypotheses from Lemmas 2 and 3. Since $Z_1, \ldots, Z_{N_1-1}$ are not independent, the joint distribution of all N_1 values of Z_i does not follow the marginal distribution.

Accordingly, the proposed test will reject the null hypothesis $\mathcal{H}_0^{(i)} : i \in P_1$ in favor of $\mathcal{H}_1^{(i)} :\ i \notin P_1$ for small values of Z_i, say if $Z_i \leq a_\alpha$ for some suitable critical value



(A) Empirical distribution of d_{IJ} for $n = 50$

(B) Empirical distribution of d_{IJ} for $n = 100$

(C) Empirical distribution of Z_i for $n = 50$

(D) Empirical distribution of Z_i for $n = 100$

Figure 2.2: Simulated data demonstrating the joint distribution of Z_i, given $N_1 = 100$ peptides in P_1 and $N_2 = 1000$ peptides in P_2. All parameters used to generate the data are the same except for the number of biological samples: $n = 50$ and $n = 100$. In (A) and (B), the distributions of d_{IJ} is given for $I, J \in P_1$ (red) and $I \in P_1, J \in P_2$ (blue) for each value of n. The black vertical lines shows t^*. In (C) and (D), the corresponding distributions of $Z_1, \ldots, Z_{N_1}$ are shown (solid red lines), along with the marginal distribution for each Z_i under both the null and alternative hypothesis.

a_α (to be explicitly determined below) which should satisfy,

$$\bar{\alpha} := \Pr\left(Z_i \leq a_\alpha \middle| \mathcal{H}_0^{(i)}\right) = \Pr\left(Z_i \leq a_\alpha | \, \tau\right) \leq \alpha, \tag{2.16}$$

for each $i = 1, \ldots, N_1$ and some fixed (and small) $\alpha \in (0, 0.5)$. The constant a_α is the (appropriately calculated) α^{th} percentile of the $Bin(N_1 - 1, \tau)$ distribution. That is, if $b(k, n, p)$ denotes the c.d.f. of a binomial (n, p)-random variable X,

$$Pr(X \leq k|n, p) := b(k, n, p) \equiv \sum_{j=0}^{k} \binom{n}{j} p^j (1-p)^{n-j},$$

then for given α and τ, the value a_α is determined so as

$$a_\alpha = \arg\max_{\ell} \left\{ b(\ell, N_1 - 1, \tau) \leq \alpha \right\}. \tag{2.17}$$

The final result of this repeated testing procedure is given by the set of all peptides in P_1 for which $\mathcal{H}_0^{(i)} : \ i \in P_1$ was rejected,

$$\mathcal{R}_\alpha := \{i; \quad i = 1, \ldots, N_1 : Z_i \leq a_\alpha\},$$

providing the set of those peptides in P_1 for which the binomial threshold is achieved and therefore have been misidentified. Similarly,

$$\mathcal{A}_\alpha := \{i; \quad i = 1, \ldots, N_1 : Z_i > a_\alpha\} = \{1, \ldots, N_1\} \setminus \mathcal{R}_\alpha,$$

provides the set of peptides correctly identified in P_1. It remains only to determine the optimal value of a_α.

2.1.2 Controlling type I errors through a_α

Let

$$\mathcal{H}_0^* = \bigcap_{i=1}^{N_1} \mathcal{H}_0^{(i)} \quad \text{and} \quad \mathcal{H}_1^* = \bigcup_{i=1}^{N_1} \mathcal{H}_1^{(i)}. \tag{2.18}$$

The hypothesis $\mathcal{H}_0^*$ above states that all the peptides in P_1 are correctly identified, whereas $\mathcal{H}_1^*$ is the hypothesis that at least one of the peptides in P_1 is misidentified.

Denote by $R = |\mathcal{R}_\alpha|$ the cardinality of the set $\mathcal{R}_\alpha$,

$$R = \sum_{i=1}^{N_1} \mathrm{I}[Z_i \leq a_\alpha]$$

so that R is a random variable taking values over $\{0, 1, \ldots, N_1\}$. Note trivially that $N_1 - R \equiv |\mathcal{A}_\alpha|$.

Consider the "global" test which rejects $\mathcal{H}_0^*$ in (2.18) if for at least one i, $i = 1, \ldots, N_1$, $Z_i \leq a_\alpha$ or equivalently, if $\{R > 0\}$. The probability of the Type I Error associated with this "global" test is therefore

$$\begin{aligned} \alpha' := \Pr\left(R > 0 \mid \mathcal{H}_0^*\right) &= \Pr\left(\bigcup_{i=1}^{N_1} \{Z_i \leq a_\alpha\} \mid \mathcal{H}_0^*\right) \\ &\leq \sum_{i=1}^{N_1} \Pr\left(Z_i \leq a_\alpha \mid \mathcal{H}_0^{(i)}\right) = N_1 \bar{\alpha} \leq N_1 \alpha, \end{aligned} \tag{2.19}$$

since by (2.16) $\bar{\alpha} \leq \alpha$. The calculations for α', can be controlled by taking $\alpha = \alpha_0/N_1$ for some α_0, to ensure that $\alpha' \leq \alpha_0$ and that $\bar{\alpha} \leq \alpha_0/N_1$.

Note that if $\{Z_i, Z_2, \ldots, Z_{N_1}\}$ were to be independent or associated (Esary et al., 1967) random variables then under $\mathcal{H}_0^*$, $\mathrm{I}[Z_i \leq a_\alpha] \sim Bin(1, \bar{\alpha})$, $i = 1, 2, \ldots, N_1$, and $R \sim Bin(N_1, \bar{\alpha})$. In this case,

$$\Pr\left(R = 0 \mid \mathcal{H}_0^*\right) = (1 - \bar{\alpha})^{N_1} \geq \left(1 - \frac{\alpha_0}{N_1}\right)^{N_1}.$$

It follows for sufficiently large N_1 (as $N_1 \to \infty$), that

$$\alpha' = 1 - \Pr(R = 0 \mid \mathcal{H}_0^*) \to 1 - e^{-\alpha_0} < \alpha_0.$$

Type II errors

The distribution of Z_i under the alternative hypothesis is explicitly available (Lemma 3), so the type II error rate of the procedure can also be explicitly controlled. For example, consider a test which rejects the alternative hypothesis $\mathcal{H}_1^{(i)} : i \notin P_1$ in favor of $\mathcal{H}_0^{(i)} : i \in P_1$ for large values of Z_i, say if $Z_i \geq b_\beta$ for some suitable critical value b_β

satisfying

$$\bar{\beta} := \Pr\left(Z_i \geq b_\beta \,\middle|\, \mathcal{H}_1^{(i)}\right) = \Pr(Z_i \geq b_\beta |\, 1-\tau) \leq \beta \tag{2.20}$$

for some fixed (and small) $\beta \in (0, 0.5)$. Similarly to a_α, the constant b_β is the (appropriately calculated) β^{th} percentile of the $Bin(N_1 - 1, 1-\tau)$ distribution.

The symmetry of $Bin(n,p)$ and $Bin(n, 1-p)$ produces the following properties relating α with β, and a_α with b_β. Specifically, the following lemmas hold (again, proofs are deferred to Appendix A).

Lemma 4. *Suppose $X \sim Bin(n,p)$ and $Y \sim Bin(n, 1-p)$. For any α, let a_α and b_α be selected according to*

$$a_\alpha := \arg\max_x \{\Pr(X \leq x |\, p) \leq \alpha\} \tag{2.21}$$

and

$$b_\alpha := \arg\min_y \{\Pr(Y \geq y |\, 1-p) \leq \alpha\} \tag{2.22}$$

Then $b_\alpha = n - a_\alpha$.

Lemma 5. *Suppose for a fixed α, the test is conducted using a cut-off $a_\alpha \geq \frac{N_1 - 1}{2}$ that satisfies (2.16). Let $\beta(\alpha)$ be the corresponding type II error of the test such that*

$$\beta(\alpha) = \Pr(Z_i \geq a_\alpha | \mathcal{H}_1^{(i)}).$$

Then $\beta(\alpha) \leq \alpha$.

Lemma 6. *Suppose for a fixed β, the test is conducted using a cut-off $b_\beta \leq \frac{N_1 - 1}{2}$ that satisfies (2.20). Let $\alpha(\beta)$ be the corresponding type I error of the test,*

$$\alpha(\beta) = \Pr(Z_i \leq b_\beta | \mathcal{H}_0^{(i)}).$$

Then $\alpha(\beta) \leq \beta$.

These lemmas collectively state that when $a_\alpha \geq \frac{N_1-1}{2}$ then $b_\alpha \leq \frac{N_1-1}{2}$, $\beta(\alpha) \leq \alpha$, such that the type I and type II error can be controlled simultaneously. Similarly, when $b_\beta \leq \frac{N_1-1}{2}$ then $a_\beta \leq \frac{N_1-1}{2}$, $\alpha(\beta) \leq \beta$. The algorithm will thus have optimal behavior when τ is sufficiently far from 0.5 to ensure that $a_\alpha \geq \frac{N_1-1}{2}$. An explicit boundary for this behavior may be determined using the following lemma:

Lemma 7. *Let a_α be defined according to (2.16) for some fixed $\alpha < 0.5$ and $\tau > \frac{1}{2}$ (Assumption 1). Let $z_\alpha = \Phi^{-1}(\alpha)$ be the α^{th} percentile of the standard normal distribution, and define*

$$\tau^* = \frac{1}{2} + \frac{1}{2}\sqrt{\frac{z_\alpha^2}{z_\alpha^2 + N_1 - 1}}. \tag{2.23}$$

Then assuming the conditions for the binomial approximation hold, (e.g. $\min(N_1\tau, N_1\tau(1-\tau)) > 5$, (Schader and Schmid, 1989)), and $\tau \geq \tau^*$, it follows that $\alpha \geq \beta$ where

$$\beta = \Pr(Z_i > a_\alpha | 1 - \tau).$$

The resulting limits for τ are shown in Figure 2.3. Clearly, as $N_1 \to \infty$, $\tau \to 0.5$. However, this convergence is extremely slow ($O(n^{-1/2})$), as even with 200 samples and $\alpha = 0.1$, $\tau > 0.55$. For small values of N_1, $\tau > .9$ may be required, especially if α is small.

2.1.3 Estimation

Note that both $\bar{F}$ and $\bar{G}$ are generally unknown (as are t^* and ψ), but can easily be estimated non-parametrically by their respective empirical CDFs, for large $N_1 \geq N_0$ and $N - N_1 \geq N_0$ respectively, for some N_0 such that $\hat{\bar{G}}$ and $1 - \hat{\bar{F}}$ can be reasonably estimated in the neighborhood around t^*. For each given peptide $i \in P_1$,

$$\hat{G}_i(t) := \frac{1}{N_1 - 1} \sum_{j \in P_1,\ j \neq i}^{N_1} \mathrm{I}[d_{ij} \leq t], \qquad \hat{F}_i(t) := \sum_{k=2}^{K} \frac{N_k}{N - N_1} \cdot \hat{F}_i^{(k)}(t) \tag{2.24}$$

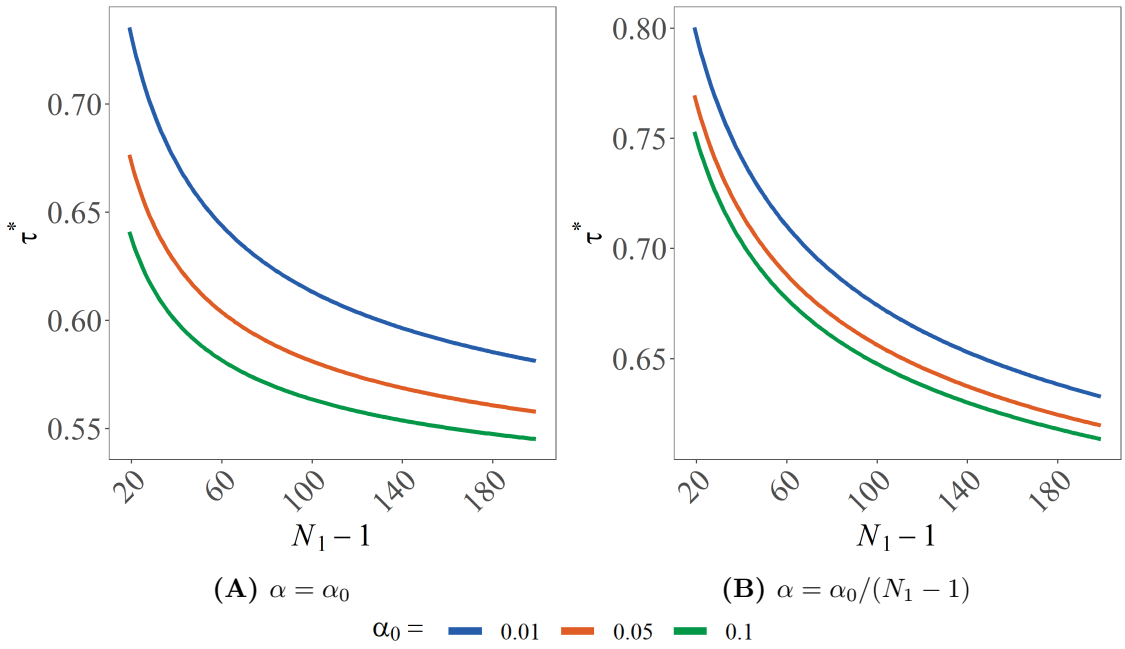


Figure 2.3: The minimum values of τ required to ensure $a_\alpha \geq 0.5$ using $\alpha = \alpha_0$ (left) and $\alpha = \alpha_0/N_1$ (right) for several values of α_0.

where, for each $k = 2, 3, \ldots, K$,

$$\hat{F}_i^{(k)}(t) := \frac{1}{N_k} \sum_{j \in P_k} \mathrm{I}[d_{ij} \leq t].$$

Clearly, $\hat{G}_i(t)$ and $\hat{F}_i(t)$ are empirical CDFs for estimating, based on the i^{th} peptide data, $\bar{G}(t)$ and $\bar{F}(t)$ respectively. Accordingly, when combined,

$$\hat{\bar{F}}(t) = \frac{1}{N_1} \sum_{i=1}^{N_1} \hat{F}_i(t), \qquad \hat{\bar{G}}(t) = \frac{1}{N_1} \sum_{i=1}^{N_1} \hat{G}_i(t), \tag{2.25}$$

are the estimators of $\bar{F}(t)$ and $\bar{G}(t)$, respectively. Further, in similarity to (2.13), set

$$\hat{\psi}(t) := \hat{\bar{G}}^{-1}(1 - \hat{\bar{F}}(t)), \tag{2.26}$$

and let $\hat{t}^*$ denote the "solution" of $\hat{\psi}(t_c^*) = t_c^*$; that is

$$\hat{t}^* := \inf_t \{\hat{\psi}(t) \leq t\}. \tag{2.27}$$

Clearly, the value of τ in (2.14) would be estimated by

$$\hat{\tau} = \hat{\bar{G}}(\hat{t}^*). \tag{2.28}$$

Note that in view of (2.24), $N_1\hat{\tau} \equiv \sum_{i=1}^{N_1} \hat{\tau}_i$, with $\hat{\tau}_i \equiv \hat{G}_i(\hat{t}^*)$ for $i = 1, 2, \ldots, N_1$. With $\hat{t}^*$ as an estimate of t^* in (2.14), $Z_i \equiv (N_1 - 1)\hat{G}_i(\hat{t}^*)$ and $\hat{\tau}_i \equiv Z_i/(N_1 - 1)$ and therefore an equivalent estimate of $\hat{\tau}$ is

$$\hat{\tau} = \frac{1}{N_1} \sum_{i=1}^{N_1} \frac{Z_i}{N_1 - 1}. \tag{2.29}$$

Lemma 8. *Under the global null hypothesis, $\hat{\tau}$ is an unbiased estimator of τ. That is,*

$$\mathrm{E}\left[\hat{\tau} | \mathcal{H}_0^*\right] = \tau$$

The proof for this lemma is given in Appendix A.

2.2 The classCleaner algorithm

In this section I present the details of a novel algorithm which I call classCleaner. The schema is shown in Algorithm 1 on page 32. Appendix contains the complete code used to implement the algorithm in R (R Core Team, 2017). As input, it takes a symmetric $N \times N$ matrix of distances, D, a vector of class labels corresponding to each row/column of the distance matrix, and a nominal desired value of α_0. The algorithm iterates over each class with at least N_0 members, where N_0 should be large enough that $\hat{\bar{G}}$ and $1 - \hat{\bar{F}}$ can be reasonably estimated in the neighborhood around t^*.

For each class k where $k = 1, \ldots, K$, the algorithm begins by calculating $\hat{\bar{G}}$ and $\hat{\bar{F}}$ according to (2.24) and (2.25). These are used to evaluate $\hat{\psi}(t)$, an estimate of $\psi(t)$ using (2.26), which is then used to find $\hat{t}^*$ and $\hat{\tau}$ using (2.26), (2.27), and (2.28). If class k has N_k elements with indices $1, \ldots, N_k$, classCleaner calculates test statistics

$Z_1, \ldots, Z_{N_k}$ according to (2.15) for each class member, using $\hat{t}^*$ in place of t^*. This test statistic is compared to a_α, which is the $\alpha = \alpha_0/N_k$.

The output of classCleaner consists of the set of indicies where the hypothesis test was rejected, and are thus likely to be incorrectly labeled. A more detailed output is also available, in which the test statistic and its probability of occurring under the null hypothesis is given. This allows the results under different values of α_0 to be compared so that an optimal values can be selected.

Data: A symmetric $N \times N$ distance matrix

Input: A vector of length N containing the assigned class for each row/column of the distance matrix.

Input: Desired type I error rate (α_0).

Result: The set of indices to be removed, $\mathcal{R}$.

set $\mathcal{R} = \emptyset$;

for *each class with size greater than* N_0 **do**

  set N_1 to be the number of instance in the class;

  obtain $\hat{\bar{G}}$;

  obtain $\hat{\bar{F}}$;

  obtain $\hat{\psi}(t^*)$;

  compute t^* such that $t^* = \hat{\psi}(t^*)$;

  compute $\hat{\tau} = \hat{\bar{G}}(t^*)$;

  set $\alpha = \alpha_0 / N_1$;

  compute a_α such that $\alpha = b(a_\alpha, N_1 - 1, \hat{\tau})$;

  for *each instance in the class* **do**

    compute $Z = \text{sum}(\text{distances within class} < t^*)$;

    if $Z < a_\alpha$ **then**
      add instance to $\mathcal{R}$

    end

  end

end

return $\mathcal{R}$;

Algorithm 1: classCleaner algorithm.

Chapter 3

Simulation Studies

To study the performance of classCleaner, I designed two simulation studies. The first simulation illustrates how classCleaner behaves over a wide range of conditions, including the number of peptides per protein, the correlation between peptides, sample size, and the proportion of misidentified peptides in the protein. In the second simulation, the focus is on evaluating the performance of classCleaner relative to other available algorithms.

3.1 Notation and methods

To establish some notation, suppose that $\mathbf{y} = (y_1, y_2, \ldots, y_N)'$ is a $N \times 1$ random vector having some joint distribution H_N. Assume, without loss of generality, that the values of $\mathbf{y}$ are standardized, so that $E(y_i) = 0$ and $V(y_i) = 1$, for each $i = 1, 2, \ldots, N$. Denote by $\mathcal{D}$ the corresponding correlation (covariance) matrix for $\mathbf{y}$, $\mathcal{D} = \text{cor}(\mathbf{y}, \mathbf{y}')$. To simplify, assume that $K = 2$ so that the N peptides are presumed to belong to either protein P_1 or P_2. Accordingly, $\mathbf{y}$ and $\mathcal{D}$ were partitioned as $\mathbf{y} = [\mathbf{y}_1', \mathbf{y}_2']'$, with, $\mathbf{y}_1 = (y_{1,1}, y_{1,2}, \ldots, y_{1,N_1})'$ and $\mathbf{y}_2 = (y_{2,N_1+1}, y_{2,N_1+2}, \ldots, y_{2,N_1+N_2})'$, $N_1 + N_2 = N$, and

$$\mathcal{D} = \begin{bmatrix} \mathcal{D}_{1,1} & \mathcal{D}_{1,2} \\ \mathcal{D}_{2,1} & \mathcal{D}_{2,2} \end{bmatrix},$$

with $\mathcal{D}_{k,\ell} = \text{cor}(\mathbf{y}_k, \mathbf{y}_\ell')$, $k, \ell = 1, 2$.

As in Chapter 2, let $\mathbf{X}$ denote the $n \times N$ data matrix of the observed intensities. Denote by $\tilde{\mathbf{x}}_j' := (x_{j1}, x_{j2}, \ldots, x_{jN})$ the j^{th} row of $\mathbf{X}$, $j = 1, 2, \ldots, n$, and assume that $\tilde{\mathbf{x}}_1', \tilde{\mathbf{x}}_2', \ldots, \tilde{\mathbf{x}}_n'$are independent and identically distributed as $\mathbf{y} \sim H_N$.

Using standard notation, write $\mathbf{1}_n = (1, 1, \ldots, 1)'$, $\mathbf{I}_n$ for the $n \times n$ identity matrix and $\mathbf{J}_n = \mathbf{1}_n \mathbf{1}_n'$ for the $n \times n$ matrix of 1s. For the simulation studiesconducted, H_N was taken to be the N-variate normal distribution, so that

$$\tilde{\mathbf{x}}_j' \sim \mathcal{N}_N(\mathbf{0},\, \mathcal{D}),\ j = 1, 2, \ldots, n, \quad i.i.d.,$$

where

$$\mathcal{D}_{1,1} = (1 - \rho_1)\, \mathbf{I}_{N_1} + \rho_1\, \mathbf{J}_{N_1} \tag{3.1}$$

$$\mathcal{D}_{2,2} = (1 - \rho_2)\, \mathbf{I}_{N_2} + \rho_2\, \mathbf{J}_{N_2} \tag{3.2}$$

$$\mathcal{D}_{2,1} = \mathcal{D}_{1,2}' = \rho_{12} \mathbf{1}_{N_2} \mathbf{1}'_{N_1} \tag{3.3}$$

for $\boldsymbol{\rho} = (\rho_1, \rho_{12}, \rho_2)$ with $0 \leq \rho_{12} \leq \rho_2 \leq \rho_1 < 1$.

To allow for misclassification of a fixed proportion p of peptides, $m := [pN_1]$ of the N_1 'observed' peptides intensities from P_1 were replaced by $\mathcal{D}^*_{1,1} = (1-\rho_2)\, \mathbf{I}_{N_1} + \rho_2\, \mathbf{J}_{N_1}$ in (3.1) above. Thus, m is the number of misclassified peptides among the N_1 peptides that were initially identified as belonging to P_1.

3.1.1 Re-paramterization of the simulation

The parameterization above is "peptide-centric", in that the values of $\boldsymbol{\rho}$ give the correlation between peptides, emphasizing the relationship of each peptide to each other peptide. An alternative "protein-centric" parameterization of $\mathbf{y}$ is also possible, which emphasizes the relationships between proteins, and on peptides within a given protein. Rewrite $\mathbf{y}$ above as

$$\mathbf{y} = \mathbf{M}\, \mathbf{w} + \mathbf{u}$$

where $\mathbf{u} = (u_1, u_2, \ldots, u_N)'$ are N i.i.d random variables from a distribution F_1 with $E[u_i] = 0$ and $V(u_i) = 1$ for each $i = 1, 2, \ldots, N$; $\mathbf{w} = (w_1, \ldots, w_K)'$ are K identically distriuted random variables from a distribution F_2 such that $\mathrm{E}[w_k] = 0$, $k = 1, \ldots, K$ and $\mathrm{var}(\mathbf{w}) = \mathbf{V}$ is a $K \times K$ positive definite matrix; and $\mathbf{M}$ is a design matrix

indicating the membership of each protein,

$$\mathbf{M}_{N\times K} = \begin{bmatrix} \mathbf{1}_{N_1} & \mathbf{0}_{N_1} & \dots & \mathbf{0}_{N_1} \\ \mathbf{0}_{N_2} & \mathbf{1}_{N_2} & & \mathbf{0}_{N_2} \\ \vdots & & \ddots & \vdots \\ \mathbf{0}_{N_K} & \dots & \dots & \mathbf{1}_{N_K} \end{bmatrix}.$$

Independence is assumed between $\mathbf{w}$ and $\mathbf{u}$. Thus, the joint distribution of $\mathbf{y}$ (denoted H_N as before) is the sum of random variables distributed F_1 and F_2, where F_1 represents the distribution of measured peptide intensities within a given protein and F_2 gives the distribution of protein intensities.

In the case of two proteins, let the correlation between the proteins be defined as λ such that

$$\mathbf{V} = \begin{pmatrix} \sigma_1^2 & \lambda\sigma_1\sigma_2 \\ \lambda\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}.$$

Then

$$E(y_{i(k)}) = E(u_i) + E(w_k) = 0 \qquad V(y_{i(k)}) = V(u_i) + V(w_k) = 1 + \sigma_k^2$$

$$\rho_1 = \mathrm{cor}(y_{i(1)}, y_{i'(1)}) = \frac{\sigma_1^2}{1+\sigma_1^2} \qquad \rho_2 = \mathrm{cor}(y_{i(2)}, y_{i'(2)}) = \frac{\sigma_2^2}{1+\sigma_2^2}$$

$$\rho_{12} = \mathrm{cor}(y_{i(1)}, y_{i'(2)}) = \frac{\lambda\sigma_1\sigma_2}{\sqrt{(1+\sigma_1^2)(1+\sigma_2^2)}}$$

where ρ_1, ρ_2, and ρ_{12} are the correlation parameters in the original parameterization. When F_1 and F_2 are normal, both parameterizations are equivalent for suitably selected values of λ, σ_1, σ_2, ρ_1, ρ_2, and ρ_{12}.

In order to fix the observed value of N_k for each protein, misidentification of peptides is modeled by randomly selecting $m := [pN_k]$ peptides from each protein and replacing the corresponding rows of $\mathbf{M}$ with $\mathbf{M}^*$, where the column containing 1 is

selected at random from the $K-1$ incorrect assignments proportionally to the number of peptides assigned to each protein.

All code required to generate data according to either simulation framework in R (R Core Team, 2017) is provided as part of the classCleaner package and is included in Appendix B.

3.2 A simulation of classCleaner with two proteins

A total of $B = 1000$ simulation runs were conducted using the test procedure described in Chapter 2, with $n = 25$, 50, 75, 100, 150, 200, 250, 300, 350, 400, 500, 600, 700, 800, 1000, $N_1 = 25$, 50, 100, 500; $N_2 = 1000$; $\alpha_0 = 0.05$; and $\boldsymbol{\rho} = (\rho_1 = 0.5, \rho_{12} = 0.1, \rho_2 = 0.5)$, $(0.5, 0.1, 0.1)$, $(0.5, 0.2, 0.5)$, and $(0.5, 0.2, 0.2)$. The value of p, the proportion of misidentified peptides was also varied such that $p = 0.0$, 0.05, 0.1, 0.2, 0.25. In particular, $p = 0$ means no mislabeling and that the initial labeling is perfect.

Remark 1. *In this simulation, ρ_2 acts as a proxy for the probability of two misidentified peptides coming from the same cluster. When $\rho_2 = \rho_1$, distances for two misidentified peptides have the same distribution as two correctly identified peptides, as would be the case for binary classification when $\rho_2 = \rho_1$. When $\rho_2 = \rho_{12}$, distances for two misidentified peptides have the same distribution as a distances between a correctly identified peptide and a misidentified peptide. This would be the case if the probability that two misidentified peptides come from the same protein is zero.*

For each simulation run, $\hat{\tau}$ and $\hat{t}^*$ were recorded, as well as the number of *true positives* (TPs), *true negatives* (TNs), *false positives* (FPs), and *false negatives* (FNs), as defined in Table 3.1. From this data, the sensitivity, specificity, FDR, *false omission rate* (FOR), and percent change in FOR (%Δ) were calculated for each run. These properties are defined as follows:

Table 3.1: For a single run, each peptide has one of four possible outcomes. The notation for the total count of peptides with each these outcomes in a single run.

		Truth		
		Correctly Identified	Misidentified	Total
Result	Keep	TN	FN	$N_1 - R$
	Remove	FP	TP	R
		$N_1 - m$	m	N_1

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$ Proportion of correctly removed peptides out of all misidentified peptides.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$ Proportion of correctly retained peptides out of all correctly identified peptides.

$$\text{FDR} = \frac{\text{FP}}{\max(\text{TP} + \text{FP}, 1)}$$ Proportion of correctly removed peptides out of all those removed.

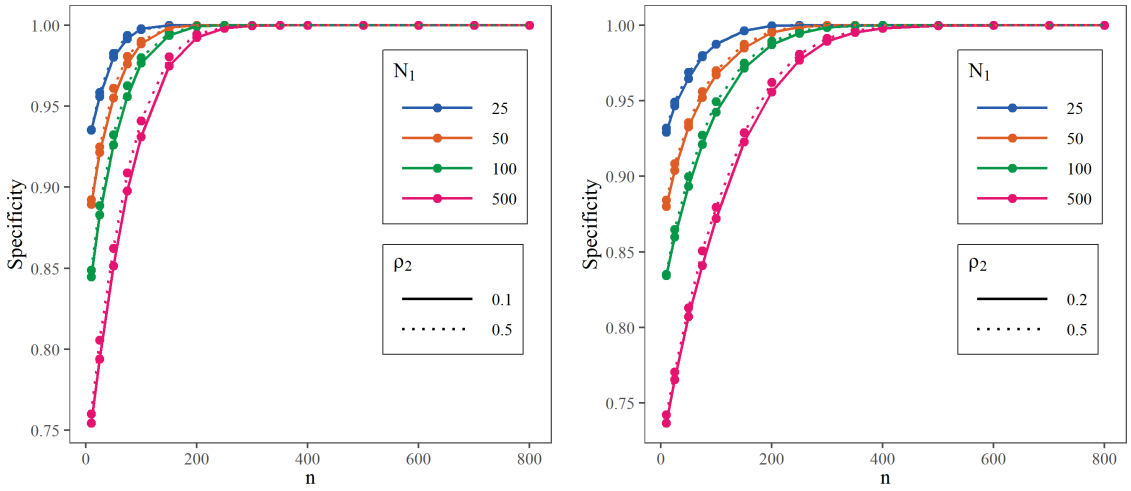
$$\text{FOR} = \frac{\text{FN}}{\max(\text{TN} + \text{FN}, 1)}$$ Proportion of correctly retained peptides out of all those retained.

$$\%\Delta = \frac{\text{FOR} - p}{p} \times 100$$ Percent change in FOR relative to p.

When $\%\Delta$ is universally less than zero, the percent reduction in FOR is defined as -$\%\Delta$. Each statistic was averaged over all 1000 runs.

3.2.1 Results with no mislabeling

When all peptides are identified correctly ($p = 0$), the global null hypothesis holds. In a practical sense, this changes the analysis in that only the specificity and FDR are defined and non-trivial, as the sensitivity is undefined and the FOR is equivalent to zero. Beyond these considerations, the primary question of interest is whether or not the theoretical assumptions hold under the conditions of the simulation.



(A) $\rho_{12} = 0.1$ **(B)** $\rho_{12} = 0.2$

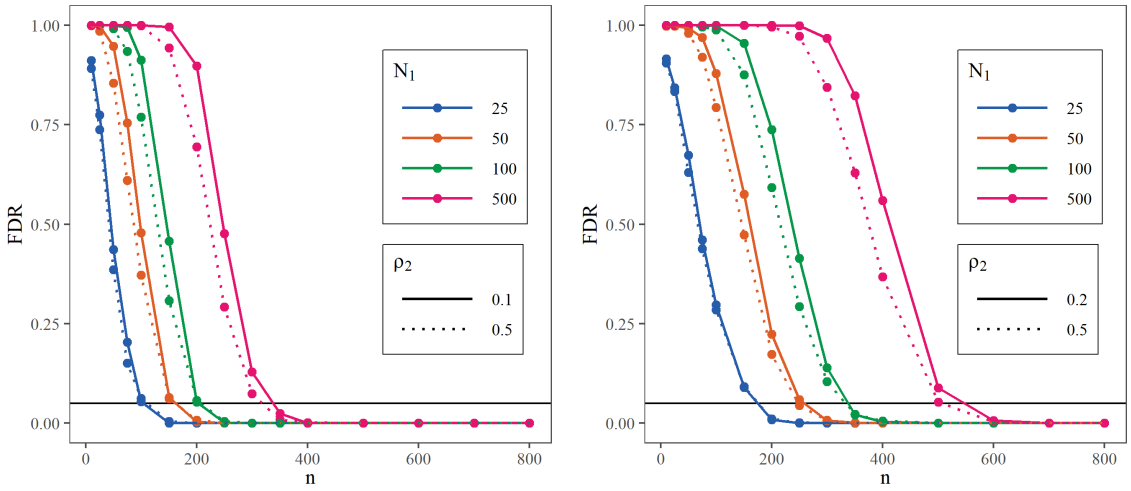
Figure 3.1: The specificity as a function of n, N_1 for (A) $\rho_{12} = 0.1$ and (B) $\rho_{12} = 0.2$ where $p = m = 0$.

The specificity of the algorithm measures its ability to keep peptides which were identified correctly. As shown in Figure 3.1 and Table 3.2 (for $\rho_{12} = 0.2$), the specificity of the algorithm increases as a function of n for each values of N_1, but the convergence is slower as N_1 increases. The specificity is slightly higher for $\rho_{12} = 0.1$ relative to $\rho_{12} = 0.2$, as expected: the later case has less separation between $\bar{G}$ and $\bar{F}$, which makes mistakes more likely. Nevertheless, even with a small sample size of 10 the average proportion of retained peptides is over 70 %. The highest percentage of retained peptides is observed when the number of peptides is smallest. At $N_1 = 25$ and $n = 10$, the average specificity for $\rho_{12} = 0.1$ was 0.935 and for $\rho_{12} = 0.2$ it was 0.929, meaning that on average over 23 peptides out of 25 were correctly retained for both values of ρ_{12}. Conversely, at $N_1 = 500$ and $n = 10$, the average specificity for $\rho_{12} = 0.1$ was 0.757 and for $\rho_{12} = 0.2$ it was 0.739, for an average of 370 - 380 peptides out of 500 retained.

The FDR measures the proportion of "discoveries" (significant hypothesis tests) which are incorrect. When all peptides were identified correctly, all discoveries are incorrect; thus for each replicate, FDR = 1 if any peptides are removed from the

Table 3.2: Simulation results for $p = 0$, $\rho_{12} = 0.2$, and $\alpha_0 = 0.05$, averaged over ρ_2. The sensitivity and FOR are excluded from the table because they are either undefined or constant when $p = 0$.

	$N_1 = 25$			$N_1 = 50$			$N_1 = 100$			$N_1 = 500$		
n	τ	FDR	Spec.	τ	FDR	Spec.	τ	FDR	Spec.	τ	FDR	Spec.
10	0.709	0.911	0.931	0.705	0.999	0.882	0.707	1.000	0.835	0.705	1.000	0.739
25	0.817	0.839	0.948	0.818	0.998	0.906	0.817	1.000	0.862	0.819	1.000	0.768
50	0.906	0.651	0.967	0.903	0.986	0.934	0.903	1.000	0.897	0.904	1.000	0.810
75	0.947	0.450	0.980	0.946	0.945	0.954	0.945	0.998	0.924	0.946	1.000	0.846
100	0.967	0.291	0.987	0.968	0.836	0.969	0.969	0.994	0.946	0.968	1.000	0.876
150	0.988	0.091	0.996	0.988	0.524	0.986	0.989	0.915	0.973	0.989	1.000	0.926
200	0.995	0.010	1.000	0.996	0.199	0.996	0.996	0.665	0.988	0.996	0.998	0.959
250	0.997	0.002	1.000	0.998	0.052	0.999	0.998	0.353	0.995	0.998	0.986	0.979
300	0.998	0.000	1.000	0.999	0.007	1.000	0.999	0.122	0.999	0.999	0.905	0.990
350	0.999	0.000	1.000	0.999	0.002	1.000	1.000	0.022	1.000	1.000	0.726	0.996
400	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.005	1.000	1.000	0.464	0.998
500	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.071	1.000
600	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.005	1.000
700	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000	1.000
800	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000	1.000



(A) $\rho_{12} = 0.1$ **(B)** $\rho_{12} = 0.2$

Figure 3.2: The FDR as a function of n, N_1 for (A) $\rho_{12} = 0.1$ and (B) $\rho_{12} = 0.2$ where $p = m = 0$.

protein. Consequently, the average FDR provides an estimate of α' in (2.19). In Figure 3.2 and Table 3.2 (for $\rho_{12} = 0.2$), the FDR converges to zero as $n \to \infty$. Convergence is fastest when N_1 and ρ_{12} are small, and slower as N_1 and ρ_{12} increase. In particular, for a fixed value of n and ρ_{12}, the FDR increases as N_1 increases. If the algorithm assumptions held, the FDR should converge below α_0 as $N_1 \to \infty$.

Recall that the classCleaner algorithm is implemented under the assumptions made in equations (2.4) and (2.6) regarding the independence and distribution of the distances d_{ij}, for $i = 1, \ldots, N_1$, $j = 1, \ldots, N$, $i \neq j$. In particular, distances in the same row (column) of the distance matrix are assumed to be independent. In actuality, distances are at best pairwise independent, and this only holds when the distribution of distances is known. This is illustrated in the difference in results for $\rho_2 = \rho_1$ and $\rho_2 = \rho_{12}$. The parameter ρ_2 determines the distances in $\mathbf{D}_{22}$, which are not used to estimate either $\hat{\bar{G}}$ or $\hat{\bar{F}}$. However, the distances in $\mathbf{D}_{12}$ are more correlated when ρ_2 is higher. As seen in Figures 3.1 and 3.2, the resulting mean specificity is lower and FDR is higher when $\rho_2 = 0.1$, presumably corresponding to an increased variance in F.

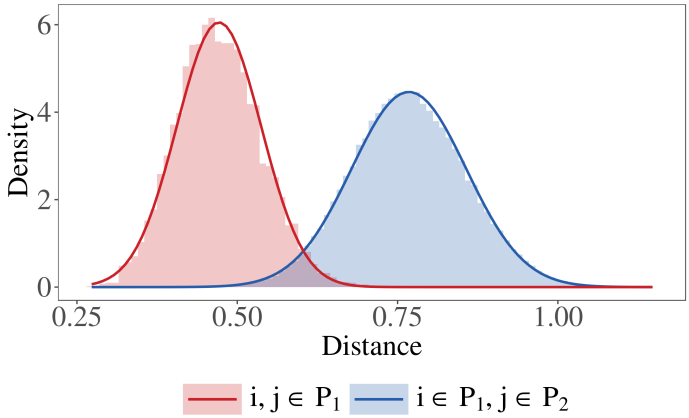


Figure 3.3: A histogram of all distances within P_1 (red) and between peptides in P_1 and those in P_2 (blue) for $p = 0$, $N_1 = 500$, $n = 100$, and $\rho_{12} = \rho_2 = 0.2$. The lines are generated from a normal distribution with mean and standard deviations matched to the data.

To verify that the FDR $\leq \alpha_0$ when the model assumptions hold, a small experiment was conducted in which the values of $\mathcal{D}$ were simulated directly. First, a distance matrix was generated using the original test procedure with $n = 100$, $p = 0$, $N_1 = 500$, $N_2 = 1000$, and $\boldsymbol{\rho} = (0.5, 0.2, 0.2)$. Normal distributions were fit to the within-P_1 distances and the P_1 to P_2 distances, as shown in Figure 3.3. For $N_1 = 25, 50, 100, 500, 1000, 2000, 5000, 12000$ and $N_2 = 1000$, these distributions were used to generate B new distance matrices by drawing d_{ij} for $1 \leq i < j \leq N$ as follows:

$$d_{ij} \sim \begin{cases} 0 & i = j \\ N(\mu = 0.523,\ \sigma = 0.0684) & 1 \leq i < j \leq N_1 \\ N(\mu = 0.771,\ \sigma = 0.0903) & 1 \leq i \leq N_1 < j \leq N \\ d_{ji} & i > j \end{cases}.$$

Five sets of $B = 1000$ runs were conducted, with the results shown in Figure 3.4. As N_1 increased, the FDR also increased, but never surpassed the theoretical limit of $1 - e^{-0.05}$. By forcing independence between distances, the theoretical behavior of the algorithm predicted mathematically holds in simulation.

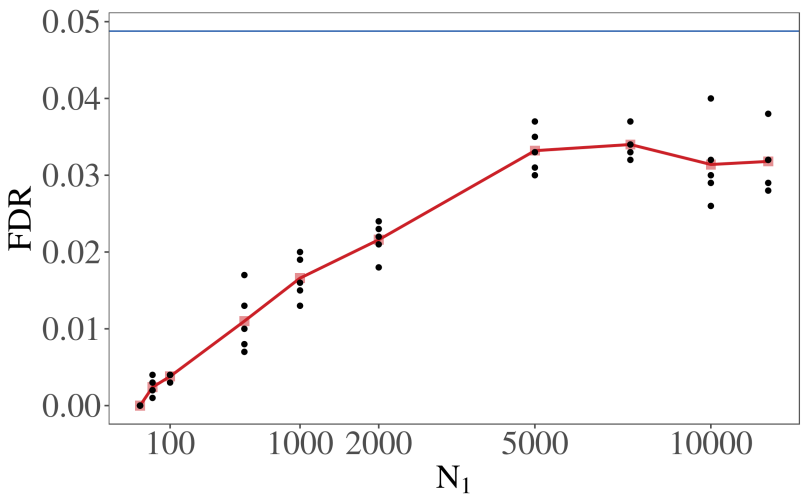


Figure 3.4: Direct simulation of the distance matrix using independent normal draws from a normal distribution and $p = 0$, performed as five batches of $B = 1000$ runs each (shown in black). The average across all five batches is shown in red. The blue line at the top of the plot shows $1 - e^{-\alpha_0}$ for $\alpha_0 = 0.05$.

For $p = 0$, the specificity and FDR both assess incorrect rejections: the specificity estimates the proportion of incorrect rejections occur in each run, while the FDR assesses the probabilty of *any* incorrect rejections. While classCleaner tends to always reject some peptides, it is most conservative when the total number of peptides is low and thus the relative importance of each peptide is highest. Where classCleaner tends to remove more peptides than desired is when the total number of peptides is high, and the relative contribution of each peptide to future models and aggregated estimates is lowest. Even in this case over 70 % of peptides were retained for $n = 10$ with better results achieved with an increase in the sample size.

3.2.2 Results for $p > 0$

Consider for a moment a trivial filtering algorithm which retains all N_1 peptides in P_1. The FOR of this algorithm is p, the specificity is 1, and the sensitivity and FDR are both 0. Note: *A FOR of p can always be achieved just by keeping the data as-is, with pN_1 mislabeled peptides.* For a non-trivial filtering algorithm to improve upon the trivial algorithm, it must result in a FOR below p. A secondary criterion for a

non-trivial filtering algorithm is to maintain a specificity as close to one as possible, measuring the ability to maximize the number of correct peptides retained. Thus, the FOR and specificity are the primary statistics on which classCleaner is evaluated.

For all parameter combinations, the FOR after applying classCleaner is smaller than p, showing that classCleaner successfully decreases the proportion of misidentified peptides in the filtered data. This can be seen for $\rho_{12} = 0.2$ in Figure 3.5, which plots the mean FOR against the mean specificity over the 1000 runs of each parameter combination. As n increases the FOR converges, with consistent results achieved by $n = 200$. For $n > 200$, the reduction in FOR relative to p was at least 99.9 % for $p = 0.05$ and all combinations of N_1, ρ_{12} and ρ_2. For higher values of p, the FOR did not converge to zero, but it was reduced by 65.9 % to 96.8 %, with larger values of N_1, ρ_{12}, and ρ_2 producing a the largest reduction. Table 3.3 shows the estimated FOR at convergence, by averaging the results from all runs where $n \geq 250$. Small sample sizes had less dramatic effects, but there was still noticeable improvement in the FOR relative to p. For $p = 0.05$ and $n = 10$, the percent reduction was 66.2 % to 86.5 %, with better results for larger values of N_1 and smaller values of ρ_{12} and ρ_2. Larger values of p lead to increased estimates of the FOR, with percent reductions only 24.4 % to 68.3 % when $p = 0.25$.

In addition to a low FOR relative to p, classCleaner also maintains a high specificity. For $n \geq 250$, the mean specificity was 1.000 for all combinations of N_1, ρ_{12} and ρ_2. The lowest mean specificity over all parameter combinations was 0.779, obtained when $N_1 = 500$, $p = 0.05$, and $\rho_2 = \rho_{12} = 0.2$. This characterizes most of the general trends over the parameters: increasing p or n increased the specificity, while increasing N_1 or ρ_{12} decreased it. The effect of ρ_2 depended on p and N_1. As stated previously, ρ_2 can influence the results of classCleaner even when $p = 0$, with a higher specificity observed for $\rho_2 = \rho_1$ relative to $\rho_2 = \rho_{12}$. For higher values of p, this is

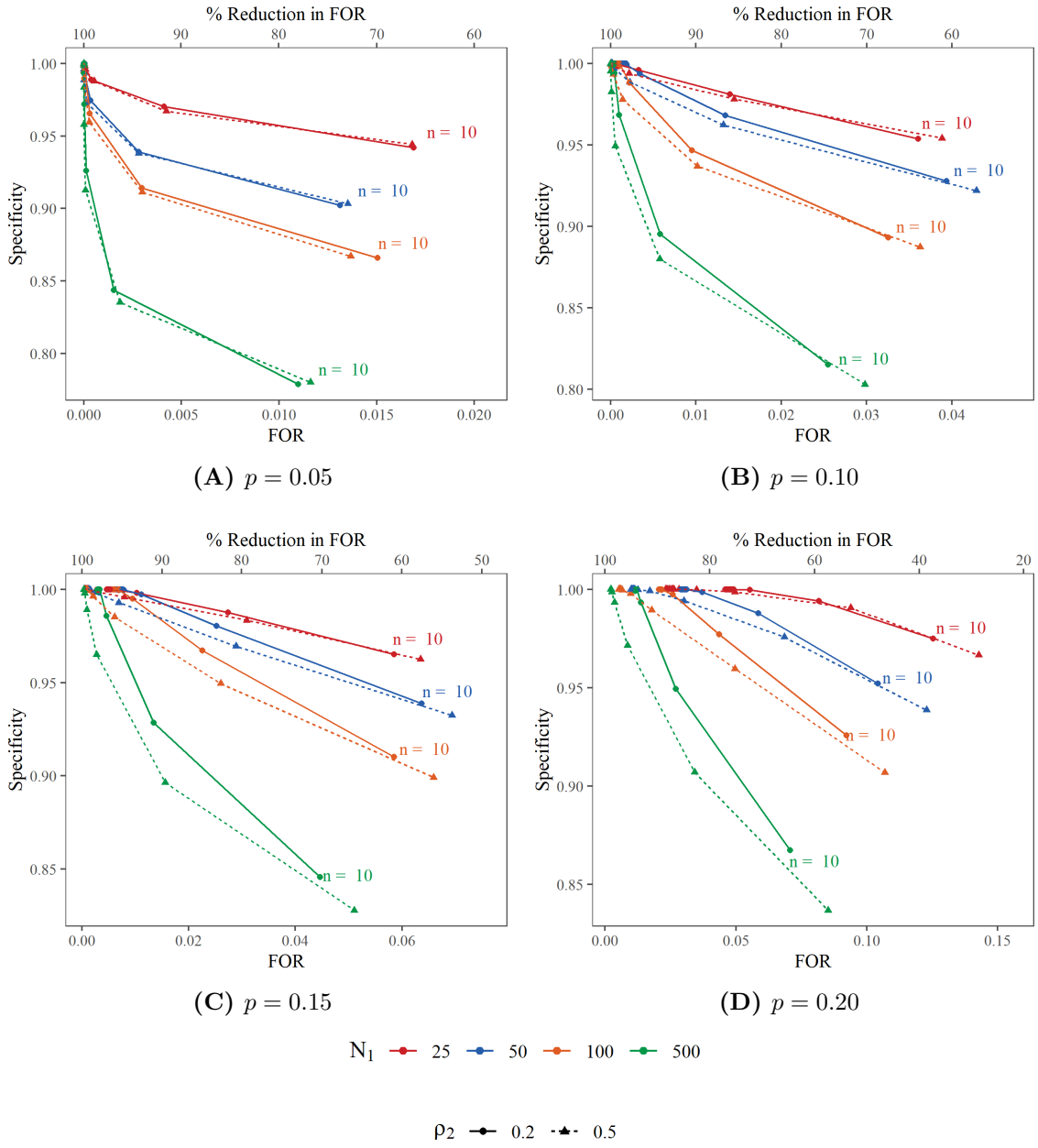


Figure 3.5: The FOR and specificity as a function of n and N_1 for $p = 0.05, 0.10, 0.15$, and 0.20 with $\rho_{12} = 0.2$. For each combination of N_1, p, and ρ_{12}, the lines show how the FOR and specificity change as n increases from $n = 10$ (labeled points) to $n = 1000$ (top left-hand corner of the plots).

Table 3.3: The mean FOR at $n = 10$ and $n > 250$ for each combination of p, N_1, and ρ_2 at $\rho_{12} = 0.2$.

		$\rho_2 = 0.2$		$\rho_2 = 0.5$	
p	N_1	$n = 10$	$n > 250$	$n = 10$	$n > 250$
0.05	25	0.0169	0.0000	0.0168	0.0000
	50	0.0131	0.0000	0.0135	0.0000
	100	0.0150	0.0000	0.0137	0.0000
	500	0.0110	0.0000	0.0116	0.0000
0.10	25	0.0361	0.0004	0.0388	0.0000
	50	0.0394	0.0015	0.0429	0.0001
	100	0.0325	0.0009	0.0363	0.0001
	500	0.0255	0.0004	0.0298	0.0000
0.15	25	0.0585	0.0050	0.0635	0.0006
	50	0.0637	0.0070	0.0694	0.0010
	100	0.0585	0.0064	0.0659	0.0009
	500	0.0446	0.0032	0.0511	0.0004
0.20	25	0.1254	0.0468	0.1427	0.0248
	50	0.1042	0.0298	0.1229	0.0107
	100	0.0922	0.0211	0.1069	0.0055
	500	0.0707	0.0116	0.0852	0.0022
0.25	25	0.1612	0.0849	0.1891	0.0683
	50	0.1399	0.0575	0.1657	0.0307
	100	0.1303	0.0485	0.1584	0.0206
	500	0.1026	0.0287	0.1292	0.0081

offset by increased bias in $\hat{\bar{F}}$ as the proportion of misidentified peptides increases. For $\rho_2 = \rho_{12}$, this effect is minimized, and it increases as ρ_2 increases.

The increase in specificity as p increases is tied to the increase in the FOR, and it is an expected consequence of the algorithm. As p increases, an increasing proportion of the distances used to approximate $\bar{G}$ and $\bar{F}$ come from other distributions, not within-P_1 distances ($\bar{G}$) or $P_1 - P_2$ distances ($\bar{F}$). This contamination affects the estimates of τ and Z_i for $i = 1, \ldots, N_1$. Ultimately, this decreases the power of the algorithm to reject (remove) peptides, increasing both the specificity and the FOR (as seen above).

Table 3.4 demonstrates $n = 50$ is sufficient to obtain a large decrease in the FOR under the model conditions. With $n = 50$, $p = 0.25$, $N_1 = 25$, and $\rho_1 = \rho_2 = 0.5$, the resulting FOR was 0.105, a decrease of 58 %. With $p = 0.15$, the FOR ranged from 0.00275 ($N_1 = 500$ and $\rho_2 = 0.5$) to 0.0111 ($N_1 = 50$, $\rho_2 = 0.2$), for a decrease of 92.6 % to 98.2 %.

Table 3.4: The FDR, FOR, sensitivity (Sens.), and specificity (Spec.) of the classCleaner algorithm using simulated data with $n = 50$ and $\rho_{12} = 0.2$.

		$\rho_2 = 0.2$					$\rho_2 = 0.5$				
p	N_1	τ	FDR	FOR	Sens.	Spec.	τ	FDR	FOR	Sens.	Spec.
0.00	25	0.902	0.673	0.000		0.965	0.909	0.630	0.000		0.969
	50	0.901	0.993	0.000		0.933	0.906	0.980	0.000		0.936
	100	0.900	1.000	0.000		0.894	0.905	1.000	0.000		0.900
	500	0.903	1.000	0.000		0.807	0.905	1.000	0.000		0.813
0.05	25	0.868	0.131	0.000	0.991	0.989	0.855	0.138	0.001	0.987	0.988
	50	0.867	0.310	0.000	0.992	0.975	0.859	0.306	0.000	0.997	0.973
	100	0.857	0.349	0.000	0.994	0.965	0.846	0.372	0.000	0.995	0.959
	500	0.859	0.542	0.000	0.998	0.926	0.846	0.559	0.000	0.999	0.913
0.10	25	0.828	0.035	0.003	0.961	0.996	0.809	0.049	0.002	0.974	0.994
	50	0.808	0.046	0.003	0.969	0.994	0.786	0.080	0.002	0.980	0.989
	100	0.810	0.086	0.002	0.980	0.988	0.788	0.144	0.001	0.988	0.978
	500	0.809	0.197	0.001	0.992	0.968	0.788	0.268	0.001	0.995	0.949
0.15	25	0.788	0.013	0.010	0.921	0.998	0.762	0.028	0.008	0.940	0.996
	50	0.772	0.016	0.011	0.930	0.997	0.745	0.040	0.007	0.957	0.993
	100	0.763	0.027	0.009	0.946	0.995	0.734	0.072	0.006	0.966	0.985
	500	0.764	0.069	0.005	0.974	0.986	0.736	0.145	0.003	0.986	0.965
0.20	25	0.720	0.001	0.055	0.760	1.000	0.682	0.009	0.050	0.784	0.999
	50	0.722	0.006	0.037	0.844	0.999	0.684	0.027	0.030	0.874	0.994
	100	0.723	0.011	0.026	0.893	0.997	0.688	0.042	0.018	0.928	0.989
	500	0.724	0.026	0.014	0.945	0.993	0.688	0.094	0.009	0.967	0.971
0.25	25	0.690	0.000	0.094	0.668	1.000	0.646	0.017	0.105	0.619	0.998
	50	0.693	0.003	0.065	0.779	0.999	0.651	0.022	0.060	0.793	0.996
	100	0.686	0.005	0.053	0.833	0.999	0.646	0.032	0.046	0.856	0.991
	500	0.687	0.013	0.031	0.903	0.996	0.647	0.067	0.022	0.934	0.976

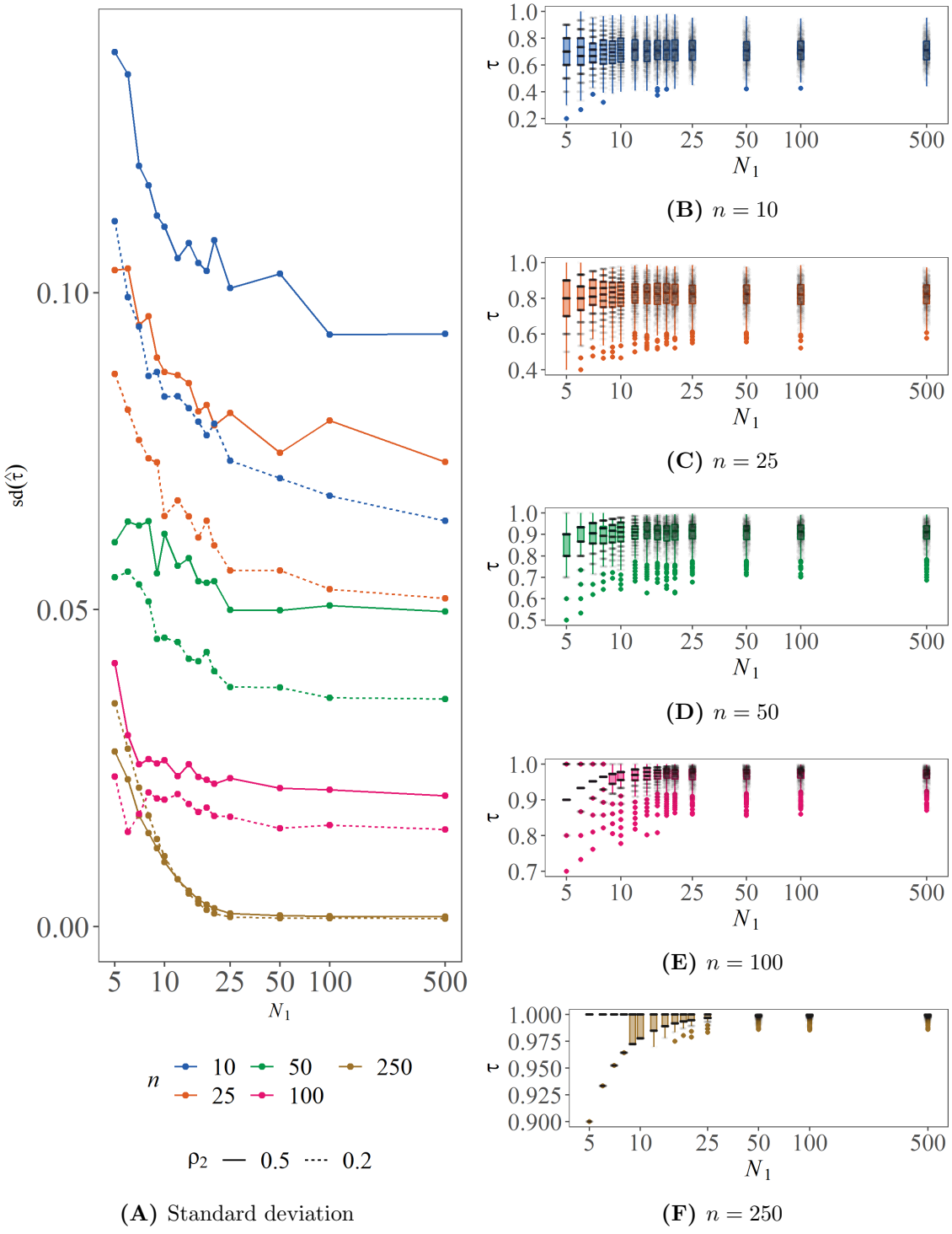


Figure 3.6: (A): The standard deviation of $\hat{\tau}$ over the $B = 1000$ simulation runs for each combination of N_1, n, and ρ_2. In (B) - (F), the distribution of $\hat{\tau}$ are shown using boxplots are shown for each value of n and $\rho_2 = 0.5$.

3.3 Evaluation of N_0 through simulation

Sections 2.1.3 and 2.2 alluded to a value N_0, below which the theoretical properties of classCleaner. This section is devoted to better illustrating the effects of using small values of N_1, and providing a recommendation on the value of N_0 which ensures that classCleaner behaves as expected.

The following simulation used the same procedure as above, with values of N_1 set to $N_1 = 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20$. Other parameter settings were as follows: $n = 10, 25, 50, 75, 100, 250$ and $\rho = (0.5, 0.2, 0.5)$ or $(0.5.0.2, 0.2)$. The value of p was set to zero, and all other parameters were unchanged from before. Results using larger values of N_1 were carried over from the previous analysis.

Figure 3.6A shows the standard deviation of $\hat{\tau}$ as a function of N_1 for each value of n and ρ_2. For simulations conducted with $n \geq 50$, the standard deviation of $\hat{\tau}$ appears to stabilize at $N_1 = 25$. The standard deviation estimates are larger and noisier for smaller values of n, suggesting that the reproducibility of the estimate is lower when n is small.

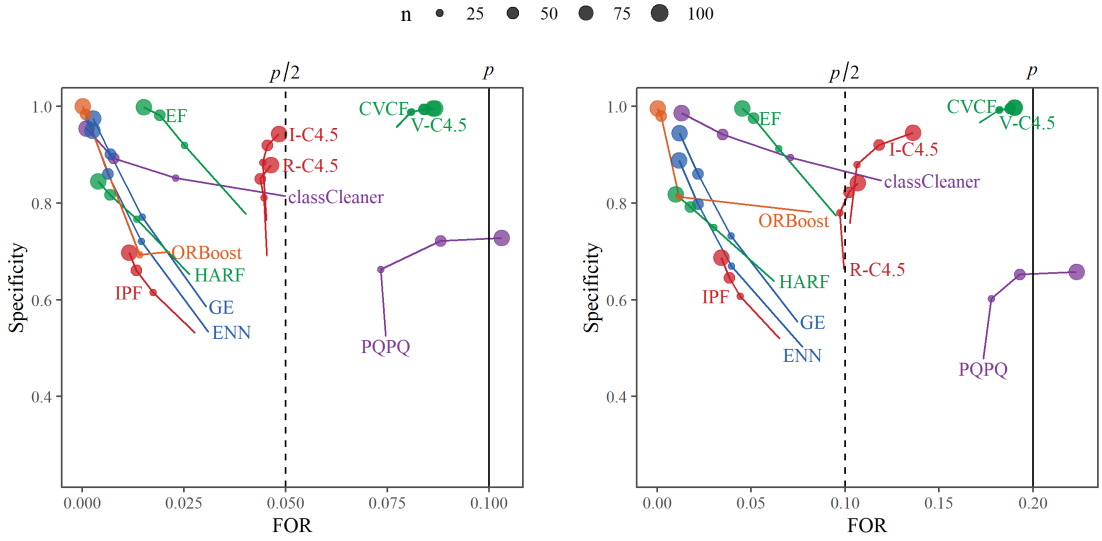
Additional clarity can be found in Figures 3.6B - 3.6F. The sparsity of the support for $\hat{\tau}$ when N_1 is small limits the precision of $\hat{\tau}$ in each simulation, and more generally in any small sample. As N_1 increases, the support becomes more dense, allowing more precision in these estimates. As n increases, τ also increases, ultimately converging to 1 (for $p = 0$).

Based on the observed simulations, taking N_0 between 20 and 25 for large values of n seems reasonable, with larger values of N_0 recommended if n is small. While this work does not directly address situations where $N_1 < N_0$, the method can be extended into this range by combining the estimates of τ and t^* from larger proteins. This will be explored in a future work.

3.4 A multi-protein simulation comparing algorithms

The final set of simulations were designed to compare classCleaner to other filtering algorithms. To this end, $n = 10,\ 25,\ 50$, or 100 independent samples from eight "proteins" were generated using the procedure described in Section 3.1.1, with $N_1 = 20,\ N_2 = 40,\ N_3 = 60,\ N_4 = 80,\ N_5 = 100,\ N_6 = 200,\ N_7 = 400$, and $N_8 = 500$ peptides in each protein, for a total of 1400 peptides. The distributions F_1 and F_2 were taken to be $F_1 = N(\mathbf{0}, \mathbf{I}_N)$, and $F_2 = N(\mathbf{0}, \mathbf{V})$ where $\mathbf{V}$ was compound symmetric with $\sigma_1^2 = \ldots = \sigma_8^2 = 1$ and $\lambda = 0.2$. This is equivalent to a peptide-centric parameterization with $\rho_1 = \rho_2 = \ldots = \rho_8 = 0.5$ and $\rho_{kk'} = 0.1$ for $k,\ k' = 1, \ldots, 8$ and $k \neq k'$. To incorporate misidentification, a total of $m_k = p \cdot N_k$ peptides were selected at random from each protein, where $p = 0.1$ or $p = 0.2$. As described in Section 3.1.1, the intensity data of these peptides was adjusted by replacing the appropriate rows of the design matrix, M with M^*, where the non-zero column was selected at random from the 7 other proteins proportionally to the number of peptides in those proteins.

Accounting for the high computational time required, the algorithm was first run on a single data set generated using $n = 25$ and $p = 0.1$. classCleaner, PQPQ, and ten other algorithms from the NoiseFiltersR package (Morales et al., 2016) which required less than 35 seconds using this small sample were used to analyze $B = 250$ simulation runs of each combination of test procedures. Each algorithm from the NoiseFiltersR package, shown in Table 3.5 was run using the default parameters. Two modifications to the default PQPQ were necessary due to differences between the simulated data and actual proteomics data: normalization was not performed, no abundance threshold was used. In addition, a confidence score of 100 was assigned to each simulated peptide.



(A) $p = 0.10$ **(B)** $p = 0.20$

Figure 3.7: The specificity and FOR for each algorithm as a function of n. At the dotted line, the proportion of misidentified peptides is 50% of that in the original data set.

3.4.1 Results

As seen in Figure 3.7 and Table 3.6, most algorithms can be divided into one of two groups based on the FOR. One group – including CVCF, R-C4.5, I-C4.5,V-C4.5, and PQPQ – have an increasing FOR as a function of n. The specificity of these algorithms varies, but tends to increase with n, with the exception of PQPQ when keeping peptides assigned to any cluster. In addition to having an increasing FOR as a function of n, the minimum FOR using these methods also tended to be high: only I-C4.5 and R-C4.5 decreased the mean specificity by at least 50 % in some cases.

Table 3.5: The 10 algorithms from the NoiseFiltersR package used in the comparative simulation study. For the complete list of all comparative algorithms, see Appendix C

Algorithms				
CVCF	EF	ENN	GE	HARF
IPF	I-C4.5	ORBoost	R-C4.5	V-C4.5

Table 3.6: The mean FOR, Specificity (Spec.), Sensitivity (Sens.) and % change in FOR (% Δ) over all 250 runs and 1400 peptides for each of 13 algorithms.

		$p = 0.1$				$p = 0.2$			
Algorithm	n	FOR	Sens.	Spec.	% Δ	FOR	Sens.	Spec.	%Δ
Proteomic									
classCleaner	10	0.050	0.615	0.815	-50.064	0.119	0.542	0.846	-40.381
	25	0.023	0.821	0.851	-77.072	0.071	0.727	0.894	-64.546
	50	0.008	0.938	0.893	-92.258	0.035	0.864	0.942	-82.626
	100	0.001	0.991	0.954	-98.971	0.013	0.948	0.986	-93.547
PQPQ	10	0.075	0.614	0.525	-25.463	0.174	0.597	0.478	-13.243
	25	0.073	0.524	0.663	-26.670	0.178	0.479	0.603	-11.079
	50	0.088	0.372	0.722	-11.935	0.193	0.378	0.653	-3.510
	100	0.103	0.249	0.728	3.104	0.223	0.246	0.657	11.658
adaBoost									
ORBoost	10	0.021	0.859	0.699	-78.605	0.082	0.656	0.782	-58.946
	25	0.014	0.968	0.693	-85.870	0.012	0.962	0.813	-94.133
	50	0.001	0.992	0.983	-99.141	0.002	0.992	0.980	-98.945
	100	0.000	0.999	1.000	-99.898	0.000	0.999	0.996	-99.882
IBL									
ENN	10	0.031	0.850	0.534	-69.077	0.077	0.834	0.502	-61.308
	25	0.014	0.906	0.721	-85.527	0.039	0.891	0.670	-80.260
	50	0.006	0.952	0.860	-93.751	0.022	0.930	0.798	-89.222
	100	0.002	0.979	0.949	-97.538	0.012	0.958	0.888	-94.201
GE	10	0.030	0.837	0.586	-69.516	0.075	0.824	0.555	-62.680
	25	0.015	0.898	0.771	-85.306	0.039	0.881	0.732	-80.349
	50	0.007	0.944	0.901	-93.093	0.022	0.924	0.860	-89.207
	100	0.003	0.976	0.975	-97.293	0.012	0.955	0.945	-94.096

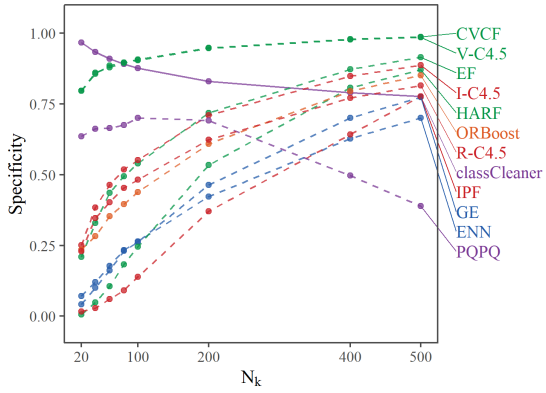
Table 3.6: *(continued)*

		$p = 0.1$				$p = 0.2$			
Algorithm	n	FOR	Sens.	Spec.	% Δ	FOR	Sens.	Spec.	%Δ
Voting									
CVCF	10	0.077	0.278	0.957	-22.735	0.172	0.196	0.968	-14.044
	25	0.081	0.216	0.988	-19.067	0.182	0.115	0.993	-8.901
	50	0.084	0.177	0.994	-15.765	0.188	0.079	0.996	-6.074
	100	0.087	0.149	0.996	-13.296	0.190	0.062	0.997	-4.825
EF	10	0.040	0.708	0.777	-59.730	0.095	0.675	0.774	-52.409
	25	0.025	0.787	0.919	-74.911	0.065	0.748	0.912	-67.676
	50	0.019	0.829	0.982	-81.008	0.051	0.789	0.976	-74.365
	100	0.015	0.861	0.998	-84.817	0.045	0.811	0.995	-77.333
HARF	10	0.026	0.843	0.652	-73.655	0.062	0.832	0.639	-68.924
	25	0.013	0.907	0.767	-86.661	0.030	0.907	0.750	-84.963
	50	0.007	0.950	0.817	-93.237	0.018	0.943	0.792	-91.129
	100	0.004	0.970	0.845	-96.123	0.010	0.968	0.818	-95.108
IPF	10	0.028	0.865	0.532	-72.289	0.065	0.857	0.520	-67.401
	25	0.017	0.903	0.615	-82.617	0.044	0.888	0.608	-77.966
	50	0.013	0.920	0.661	-86.706	0.038	0.897	0.646	-80.871
	100	0.012	0.927	0.697	-88.492	0.034	0.903	0.687	-82.928
V-C4.5	10	0.077	0.279	0.956	-22.798	0.172	0.199	0.966	-14.243
	25	0.081	0.220	0.988	-19.407	0.182	0.118	0.992	-9.094
	50	0.084	0.180	0.994	-16.051	0.188	0.079	0.996	-6.053
	100	0.086	0.155	0.996	-13.837	0.190	0.064	0.997	-4.949
Iterative									
I-C4.5	10	0.045	0.674	0.764	-54.710	0.103	0.654	0.758	-48.713
	25	0.044	0.631	0.884	-55.699	0.106	0.580	0.880	-46.887
	50	0.045	0.605	0.919	-54.537	0.118	0.506	0.920	-40.968
	100	0.048	0.568	0.943	-51.659	0.136	0.403	0.945	-31.961
R-C4.5	10	0.045	0.704	0.691	-54.630	0.099	0.707	0.664	-50.316
	25	0.045	0.657	0.811	-55.346	0.097	0.662	0.780	-51.385
	50	0.044	0.649	0.850	-56.306	0.102	0.624	0.822	-49.016
	100	0.046	0.613	0.878	-53.578	0.107	0.595	0.841	-46.580

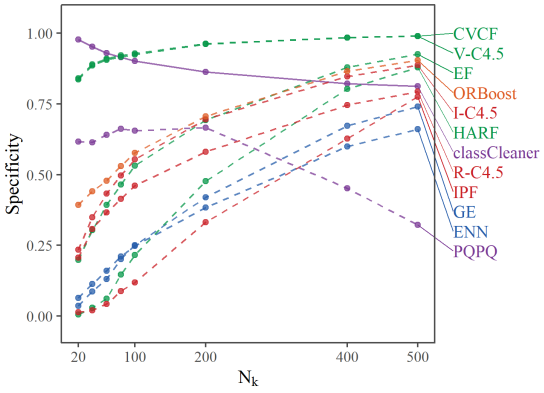
The second group includes classCleaner, EF, ENN, GE, HARF, IPF, and ORBoost. Using these algorithms, the FOR is typically low, and decreases as a function of n. For small values of n, several of these algorithms have very low specificities, with over 40 % of correctly identified peptides removed using ENN, GE, and IPF at $n = 10$. classCleaner stands out among these algorithms, because it achieves a high specificity for all values of n, with over 81 % of correctly identified peptides retained even when $n = 10$. While classCleaner is not as successful in lowering the FOR for low values of n, this drops quickly as n increases, with 93.5 % to 98.9 % fewer misidentified peptides in the filtered data set relative to the original at $n = 100$.

What distinguishes classCleaner from other filtering algorithms is its specificity as a function of protein size. As seen in Figure 3.8, most algorithms increase in specificity as N_k increases. When n is small, many of these algorithms remove almost all peptides associated with smaller proteins. In contrast, classCleaner has the highest specificity when N_k is small, even when $n = 10$. For larger values of n, e.g. $n = 100$, algorithms EF, ORBoost, V-C4.5, and CVCF had near perfect specificity across all values of N_k, but only classCleaner maintained a high specificity irrespective of n.

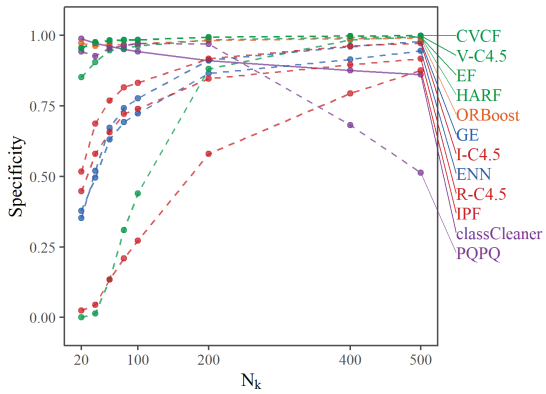
In Figure 3.9 the FOR is examined as a function of protein size as well. Here as well, the behavior of classCleaner contrasted with the behavior of the remaining algorithms. classCleaner had the highest FOR when N_k is small, with decreasing FOR as a function of protein size, all other algorithms had an increasing FOR as a function of protein size. Importantly, for large values of n, the results of classCleaner were comparable to most other algorithms. For small values of n, classCleaner had a comparable performance to other algorithms for large values of N_k. While classCleaner did produce the highest FOR for low n and low N_k, it is important to realize that many of the algorithms achieved a low FOR by removing almost all peptides from the smaller proteins. The conservative behavior of classCleaner in this region is thus a preferred outcome in this case.



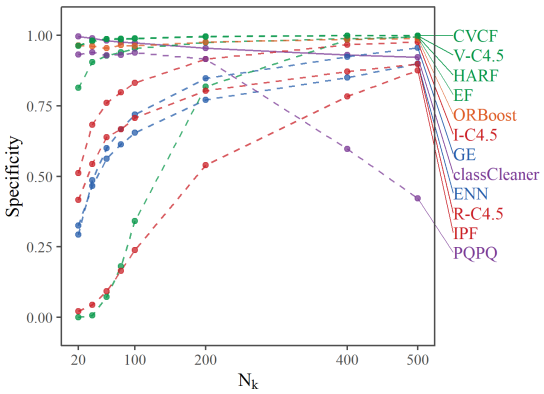
(A) $n = 10,\ p = 0.10$



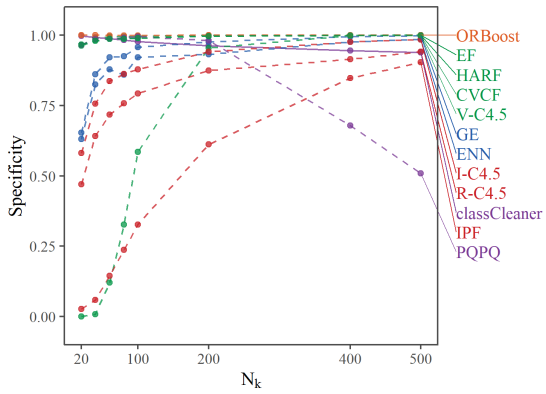
(B) $n = 10,\ p = 0.20$



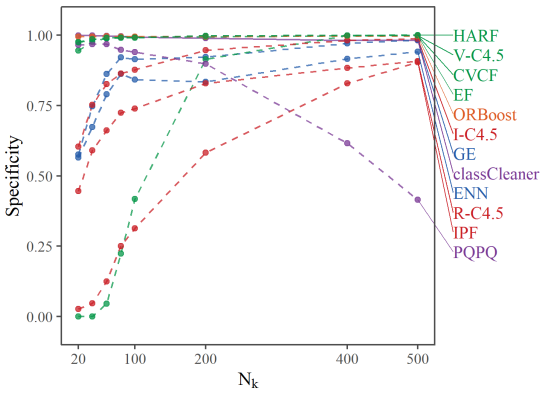
(C) $n = 50,\ p = 0.10$



(D) $n = 50,\ p = 0.20$

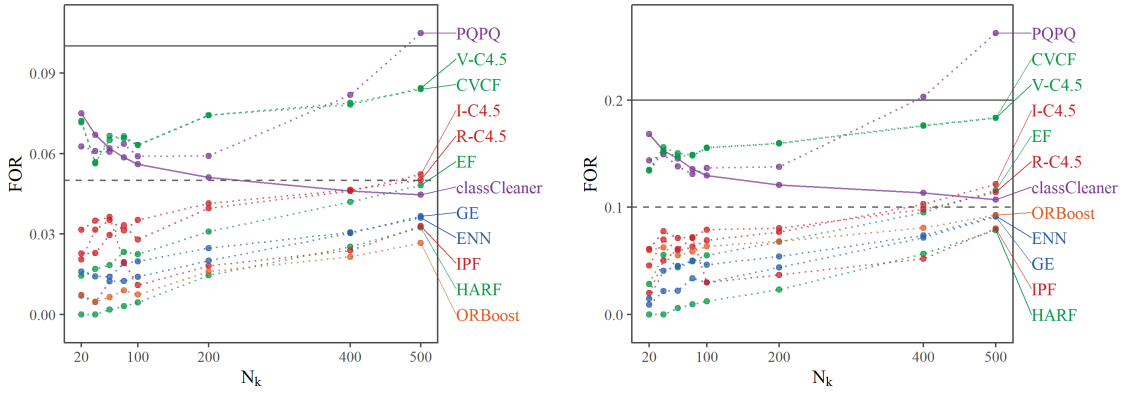


(E) $n = 100,\ p = 0.10$



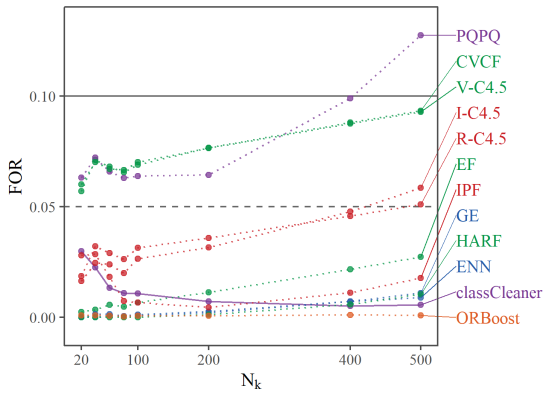
(F) $n = 100,\ p = 0.20$

Figure 3.8: Specificity of each algorithm as a function of protein size.

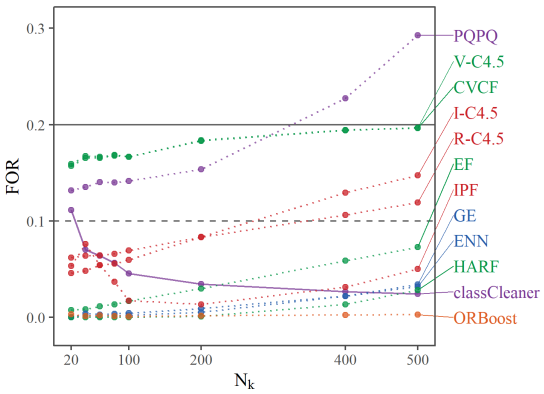


(A) $n = 10,\ p = 0.10$

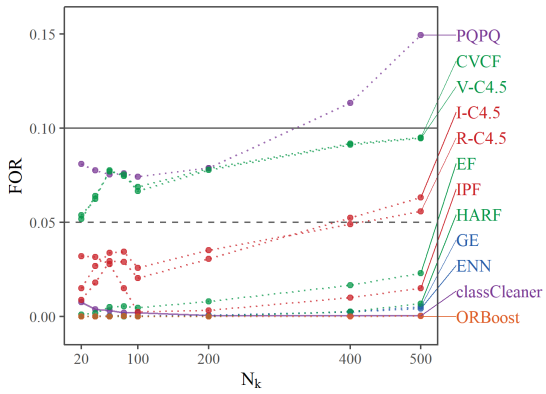
(B) $n = 10,\ p = 0.20$



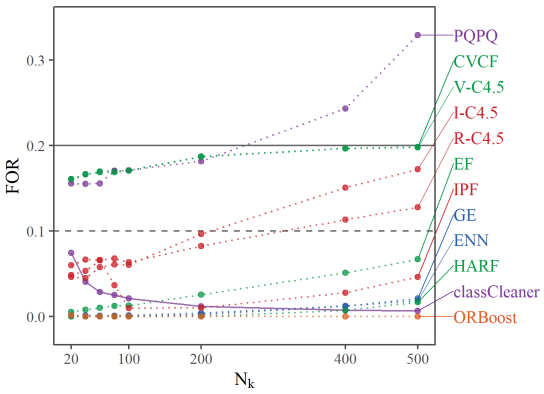
(C) $n = 50,\ p = 0.10$



(D) $n = 50,\ p = 0.20$



(E) $n = 100,\ p = 0.10$



(F) $n = 100,\ p = 0.20$

Figure 3.9: FOR of each algorithm as a function of protein size.

3.5 Conclusions

In all simulations, classCleaner was shown to improve the FOR relative to p while maintaining a high level of specificity. The algorithm works best for large sample sizes, but as shown, it can provide improvement even when $n = 10$.

Of the other algorithms compared here, results varied significantly, although most did result in a lower FOR relative to not filtering. CVCF and V-C4.5 were extremely conservative, with almost all peptides retained. HARF and IPF tended to remove the most peptides, with a resulting low specificity and FOR, however these algorithms were far more aggressive in removing peptides from smaller proteins, and thus should be used with caution in proteomics setting. The results of most other algorithms depended heavily on the sample size. For $n = 100$, multiple algorithms were able to provide a high specificity and low FOR, with classCleaner, EF, and ORBoost providing the best results. Conversely, for small values of n, both EF and ORBoost had a low specificity and low FOR consistent with removing almost all peptides from small proteins. While PQPQ did not perform as well as many other algorithms, confidence scores play an important role in determining which peptides should be used as model peptides in the actual algorithm.

Chapter 4

Application of classCleaner on a LC-MS/MS data set

4.1 Introduction

To illustrate the use of classCleaner on a real data set, it was applied to a large observational study on sickle cell disease. The study itself consists of measured peptide intensities from 120 serum samples divided into one of four groups depending on disease status. To substantiate classCleaner's ability to detect misidentified and mis-quantified peptides, sixteen additional injections were included to experimentally determine whether each peptide was quantified accurately and, for five specific proteins, whether they were identified correctly. These extra injections were divided into two subsets.

The *spike-in* subset contained eight total aliquots: five in which a single protein was spiked in at high intensity and three controls. Correctly identified peptides from the spiked proteins had an artificially high intensity in the corresponding aliquot. Misidentified peptides will lack this high intensity, and thus appear "unspiked". This information was used to objectively mark peptides with a high probability of having been misidentified.

In the *loading* subset, four samples out of the 120 experimental samples were split into three aliquots each: a standard aliquot run using the same procedure as the remaining experimental samples, and two aliquots in which the quantity of protein loaded into the mass spectrometer was varied. This is modeled on the work of Liu et al. (2009), which shows that varying the quantity of protein injected into the mass spectrometer can assist in determining the linear range of each peptide. When the measured intensity of a peptide is in the linear range for all three samples, nor-

malization removed the effect of varying the loading quantity, and the normalized intensities from the same sample were approximately the same. Peptides for which the measured intensity is outside the linear range tended to appear different even after normalization, indicating which peptides are more likely to be mis-quantified.

Neither of these subsets provide a "gold-standard" in determining the usefulness of each peptide in quantifying protein abundance. The commercially produced proteins used in the spike-in subset are likely to be less variable in their sequence and *post-translational modifications* (PTMs) than what is present in human populations, such that some correctly identified peptide sequences present in the experimental data may not be high in the spiked sample. Peptide abundance may be noisy for reasons unrelated to identification or low signal-to-noise ratios. This data is thus an independent assessment of the accuracy of the protein labels and quantification, but it is subject to the same technical sources variation as the experimental injections.

Despite these potential limitations, the presence of these subsets provides an experimental control which can be used to assess the accuracy of the identification and quantification of each peptide. Furthermore, they provide the means of creating objective, albeit imperfect, benchmarks against which the performance of classCleaner can be compared with that of other algorithms.

4.2 Study Design and methods

Each experimental sample was depleted of albumin and IgG using the ProtePrep Immunoaffinity Albumin & IgG Depletion Kit (Sigma-Aldrich) per the manufacturer's instructions. Briefly, 25 µL of sample was diluted with 75 µL equilibrium buffer and added to the top of a packed medium bed. The sample was incubated at room temperature for 5-10 minutes, then a centrifuge was used to collect the depleted sample. The sample was then sent through the column a second time using the same

Table 4.1: The original quantity of the five spike-in proteins in 25 mL of each sample in the spike-in subset, as measured by nephelometry, as well as the quantity of protein added to generate the spike sample.

	Pre-depletion (µg/25µL)			Spike Quantity (µg/75µL)
Symbol	s1	s2	s3	
A1AG1	5.53	21.18	29.50	125
APOA1	20.23	30.00	41.25	100
APOB	9.30	10.50	24.75	167
CERU	3.00	7.13	6.03	38
HEMO	9.48	20.85	22.38	68

incubation and centrifugation steps, after which 125 µL of the equilibrium buffer was added to the column to wash any remaining unbound protein.

The spike-in subset was obtained using three biological samples of EDTA plasma for which the concentration of all five spike proteins had been previously measured using nephelometry. Protein powder from *alpha-1-acid glycoprotein 1* (A1AG1), *apolipoprotein A1* (APOA1), *apolipoprotein B* (APOB), *ceruloplasmin* (CERU), and *hemopexin* (HEMO) was provided by the Ragg laboratory. One of the three biological samples (designated "s1") was split into six aliquots of 25 µL prior to depletion. A single 25 µL aliquot was obtained from the remaining two samples ("s2" and "s3"). One aliquot from each biological sample was depleted using the standard depletion protocol described above. For the remaining five aliquots (all from "s1"), between 75 µl and 550 µl of equilibrium buffer from the depletion kit was added to each tube of protein powder. The five remaining aliquots were then diluted using 75 µL of spiked equilibrium buffer, such that each aliquot was spiked with exactly one protein. The remaining steps in the depletion protocol were followed as per the manufacturer's instructions as described above. The original concentration of each sample, as well as the concentration in the spiked buffer are shown in Table 4.1.

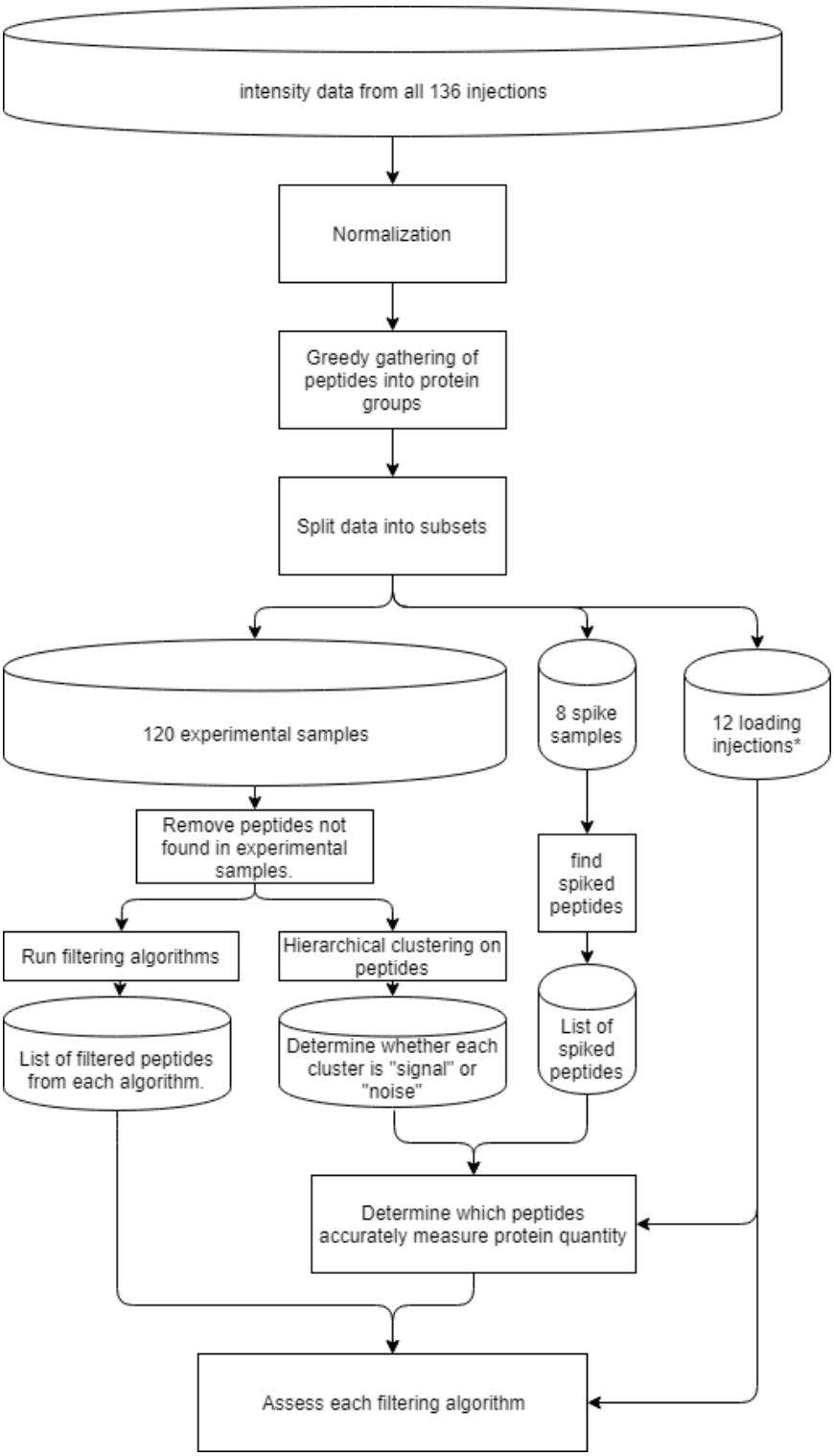


Figure 4.1: The work flow used to analyze the spike-in and loading subsets. *The 12 dilution injections include four injections which are also part of the 120 experimental samples, plus eight injections of the same samples with adjusted protein quantities.

After depletion, the 120 experimental samples and eight spike samples were sent to the The Proteomics Core at the Indiana University School of Medicine, for trypsin digestion and LC-MS/MS analysis. Processing was performed in two batches of 64 samples, and each batch was further subdivided into five groups of ten samples and two groups of twelve samples. The four samples with the lowest overall protein concentration after depletion, as measured by Bradford assay, were selected for the loading subset. These samples were split into three aliquots after trypsin digestion, and the loading quantities were varied as follows:

- The normal aliquot consisted of 80 µL of sample, of which 35 µL was injected into the mass spectrometer.
- The "low concentration" aliquot consisted of diluting 20 µL with 80 µL buffer, of which 35 µL was injected into the mass spectrometer.
- The "high concentration" aliquot consisted of 80 µL of sample, of which 55 µL was injected into the mass spectrometer.

The "low-concentration" aliquot had 20% the dilution strength of the normal aliquot, while the "high-concentration" aliquot had 157% the dilution strength of the normal aliquot. Loading and spike-in injections were all loaded into the mass spectrometer as part of the last group in the second batch, all experimental samples (including the standard aliquot of the loading subset) were allocated to the remaining groups at random.

Identification was performed using X!Tandem (Craig and Beavis, 2004) as part of the Trans-Proteomic Pipeline (Keller et al., 2005) and the UniProt Human database (The UniProt Consortium, 2017). Alignment and quantification were performed using IdentiQuantXL (Lai et al., 2011).

An overview of the data processing can be seen in Figure 4.1. Variance stabilizing normalization (Hubert et al., 2008) was performed on the entire data set. To address

the identification of peptides shared by multiple proteins, proteins which share at least one discovered peptide were combined into groups, and filtering was performed at the group level. The samples were split into the experimental, spike-in, and loading subsets after all normalization and grouping had been completed.

4.2.1 Determining misidentified and mis-quantified peptides using the validation subsets

Three decision rules were used to distinguish between peptides which are accurately identified and accurately quantified from those which are not. The first uses the difference in intensity in the spiked samples relative to the remaining samples without the spike to determine which peptides are accurately identified. The second rule uses hierarchical clustering on the experimental subset to group peptides with similar relative intensities over the samples. Peptides which did not follow the pattern or patterns expressed by most peptides tended to get included in a separate group, whether the non-conformity was due to misidentification or poor quantification. The loading subset was used to evaluate the number of clusters to create from hierarchical clustering dendrogram in order to separate out the non-conforming cluster. Finally, the results of these two decision rules were combined to create a third determiner of the accuracy of each peptide in measuring the relative abundance of each sample. The results of these decision rules are three benchmarks, labeled the Spike, *hierarchical clustering* (HC), and Hybrid.

Spike-in

For each spiked protein P, the eight intensities for each peptide were centered by subtracting out the median of the five intensities from the "s1" sample *not* spiked with P. The empirical distribution of each of the five unspiked "s1" samples was calculated using the centered intensities. These values were averaged to create a

reference distribution for the typical behavior of peptides from P in the "s1" sample. The quantile of each centered intensity in the spiked "s1" sample was then calculated against the reference distribution. Quantiles above 0.975 were assumed to be from the spiked protein and correctly identified. Quantiles between 0.005 and 0.975 were assumed to be from misidentified peptides. Quantiles less than 0.005, were considered to be invalid/missing and were ignored when assessing the peptides using only the spike-in data.

Hierarchical Clustering and hybrid

The intensities of each peptide in the experimental and loading subsets were (separately) centered and scaled to have a mean of zero and variance of one. Using the standardized intensities from the experimental data set, a dendrogram were created for each protein using hierarchical clustering with Euclidean distance and the Ward method of agglomerative clustering (Ward, 1963). To determine the optimal number of clusters, M^*, for each protein, groups created by splitting the dendrogram into $M = 1, \ldots, 8$ total clusters were considered. Because increasing the total number of clusters from M to $M+1$ causes exactly one parent cluster to be split into two child clusters, studying one to eight total clusters results in 15 distinct clusters to examine for each protein.

For each cluster, the corresponding peptide intensities in the *loading* subset were used to fit the following model:

$$y_{ijk} = \mu_i + \epsilon_{ijk}$$

where y_{ijk} is the standardized intensity of the k^{th} observation ($k = 1, 2, 3$, corresponding to the measurements at each loading quantity) from the i^{th} sample ($i = 1, 2, 3, 4$), and j^{th} peptide in the cluster of interest, μ_i is the average intensity for sample i, and ϵ_{ijk} is random error. The *mean squared error* (MSE) of this model was obtained and

used to classify the proposed cluster into one of three groups. Clusters with a MSE above 0.75 were considered noisy, MSEs below 0.40 were considered clean, and those with a MSE between 0.4 and 0.75 were considered intermediate. For $M \geq 2$, the MSEs of the two newly created child clusters were compared to that of their parent cluster. Values of M such that both child clusters had at least ten peptides and at least one new cluster had a MSE in a different category (clean, intermediate, or noisy) were added to a set, designated $\mathcal{M}$.

To find M^*, only cluster totals in $\mathcal{M}$ were considered. For each $M \in \mathcal{M}$, the HC and hybrid decision rules were evaluated over all all peptides. Using the HC rule, all peptides in noisy clusters were classified as bad (misidentified or inaccurately quantified) while peptides in intermediate or clean clusters were classified as good (correclty identified and accurately quantified). The hybrid rule follows the HC rule for noisy and clean clusters, but treats unspiked peptides in intermediate clusters as bad. Cluster totals $M_1 < M_2$ are defined to be equivalent if they produce the same outcome across all peptides – that is, the same peptides are marked good and bad across for both M_1 and M_2. The value of M^* was found by taking the smallest $M \in \mathcal{M}$ such that M^* is equivalent to the largest M in $\mathcal{M}$.

The result of these decision rules are three *benchmarks* based on experimental data which can be used to determine whether each peptide should be retained or removed. These benchmarks are used in place of the unknown truth for the purpose of estimating the specificity, sensitivity, FOR, FDR, and $\%\Delta$ of each algorithm.

4.2.2 Filtering algorithms

Filtering algorithms were applied to peptides identified in at least one experimental sample; peptides identified only in the loading or spike-in subsets were removed. Furthermore, all peptides mapped to groups with no reviewed (Swiss-Prot) entries

Table 4.2: The 14 algorithms used in the LC-MS/MS example. For the complete list of all comparative algorithms, see Appendix C.

Algorithms				
BBNR	CVCF	EB	EF	ENG
ENN	GE	HARF	I-C4.5	ORBoost
PF	PQPQ	R-C4.5	V-C4.5	

in the UniProt database and those with only a single identified peptide were also removed.

A list of all filtering algorithms applied to the experimental data set is shown in 4.2. Other than PQPQ and ORBoost, algorithms were run with the default parameters. PQPQ was run without the default median-normalization, since normalization had already been done, and only peptides assigned to the first cluster were retained. The number of boosting iterations used for the OR Boost algorithm was set to six (6) instead of 20 because that was the last iteration that reduced the size of the data set without removing every peptide. Distances in the classCleaner algorithm were calculated correlation distance (Eq. 2.2) and α_0 was set to 0.05. Peptide reclassification (e.g. replacing the protein identity with a different one), as performed by the GE algorithm was ignored, and these peptides were treated as incorrect.

The input for BP-Quant is the set of results from contrasts between the groups of interest obtained by analyzing each peptide separately. For this analysis, the six contrasts were obtained by comparing the four groups in the experimental data in a pairwise fashion using an *analysis of variance* (ANOVA) model. For each contrast and protein, a *Benjamini-Hochberg* (BH) correction (Benjamini and Hochberg, 1995) was applied to the p-values across all peptides. The p-values were then discretized to $\{-1, 0, 1\}$ where zero corresponds to a non-significant result while -1 and 1 correspond to significant negative and positive results, respectively. The expected frequency for peptides with no significant results for all six contrasts was set to $(1-0.05)^6 = 0.7351$.

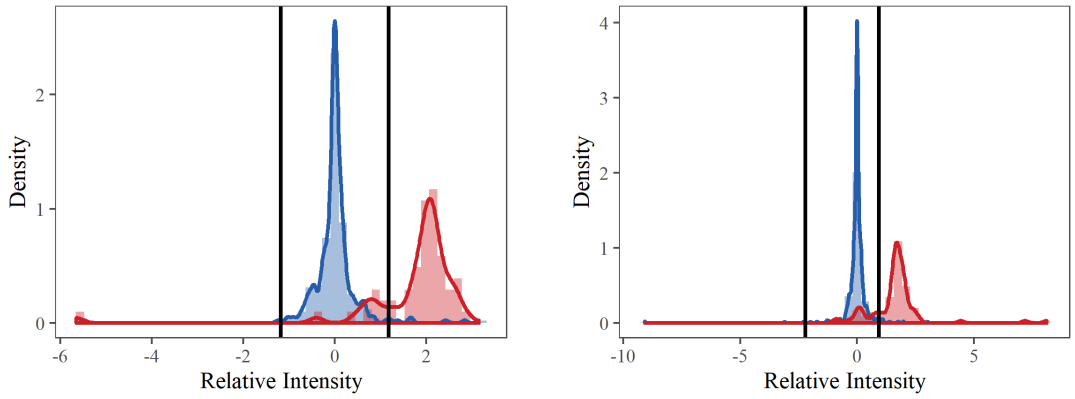
4.3 Results

In total, 8219 unique peptides were identified in at least one of the 136 injections. Of these, 656 were removed because they were not identified any of the 120 experimental samples. A further 99 peptides mapped to protein groups without an associated SwissProt ID (i.e. only unreviewed TrEMBL sequences). Of the remaining 7464 identified peptides included in the analysis, 886 peptide were identified as belonging to one of the five spiked proteins.

All peptides identified as belonging to APOA1, APOB, CERU, or HEMO mapped to only a single reviewed sequence (the target protein). Of the 37 peptides identified as belonging to A1AG1, seven also mapped to *alpha-1-acid glycoprotein 2* (A1AG2), and an addition 20 peptides were identified as belonging to only A1AG2. Together, these peptides formed the *alpha-1-acid glycoprotein 1 & 2* (A1AG1-A1AG2) protein group.

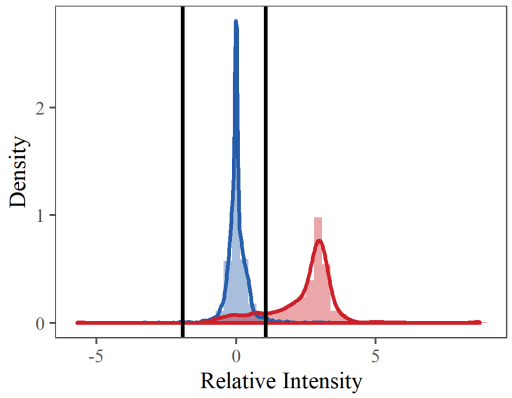
4.3.1 Hierarchical clustering and spike-in results

Figure 4.2 compares the estimated density of centered intensities from the spike-in aliquot (red) with those obtained using the five other "s1" aliquots (blue). The mean intensities from the spiked aliquots were right-shifted by 3.16 to 6.34 standard deviations relative to the corresponding means of the corresponding peptides from the non-spiked samples. Intensities from the spiked sample from all five proteins showed features consistent with the possibility of misidentification. The clearest evidence was from APOA1 and CERU, which show bimodal distributions with small peaks centered at zero. A1AG1-A1AG2 also had a bimodal distribution, although the smaller peak was centered to the right of zero. APOB had a long left tail in the region where 95 % of the intensities from non-spiked samples fell, but no peak was visible. Finally, HEMO only had six peptides with intensities below the 97.5^{th} percentile of the non-spiked distribution, which was insufficient to draw any conclusions. A total of three peptides

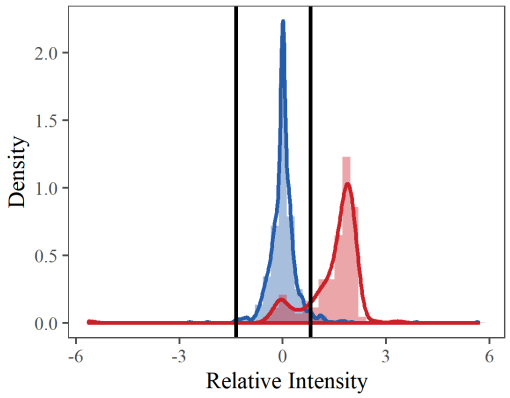


(A) A1AG1, A1AG2

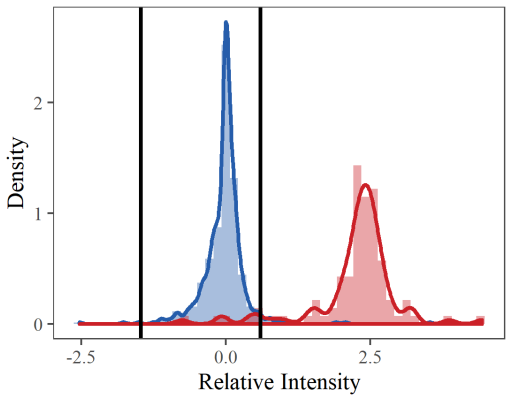
(B) APOA1



(C) APOB

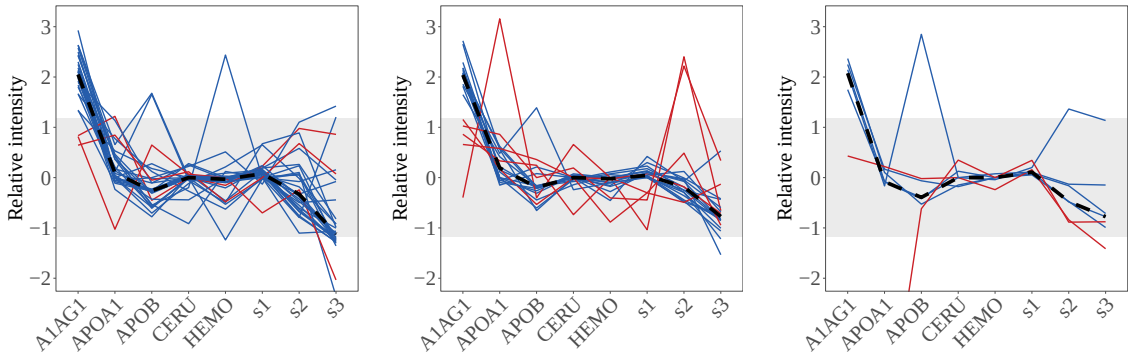


(D) CERU



(E) HEMO

Figure 4.2: The centered intensities of each peptide using the spiked (red) and non-spiked (blue) samples from each protein. The vertical lines show the 0.5^{th} and 97.5^{th} percentiles of the distribution of unspiked samples.



(A) A1AG1 **(B)** A1AG2 **(C)** Shared

Figure 4.3: The centered intensities of each peptide in the A1AG1-A1AG2 protein group are shown for each aliquot, divided into peptides belonging only to A1AG1 (A), only belonging to A1AG2 (B), and those shared between the two proteins (C). Peptides shown in red fell below the 97.5^{th} percentile of the non-spiked aliquot, and thus were determined to be unspiked. Aliquots are labeled according to the spike protein added to the s1 sample, aliquots with no spiked protein are labeled according to the originating sample.

were removed due to extremely low intensities (the intensity from the spiked aliquot fell below the 0.5^{th} percentile of the non-spiked aliquots). Of the 20 peptides identified as belonging to A1AG2 but not A1AG1, 15 had centered intensities consistent with the spike signal (Figure 4.3). Three of the peptides from A1AG1 and one of the shared peptides also appear unspiked, while one additional shared peptide was missing in the spiked sample.

Using hierarchical clustering, the peptides from each protein were split into two to four clusters, as seen described in Table 4.3. Peptides belonging to HEMO were split into two clusters, shown in Figure 4.4. Cluster 1, on the bottom, contained 70 spiked peptides with a MSE of 0.345 on the loading subset, suggesting that these peptides are accurately identified and accurately quantified. In the figure, these peptides are low (blue) in samples displayed in the left-most columns and high (red) in samples displayed to the right, supporting the "clean" classification based on the loading subset. Cluster 2, on the top, contains 22 spiked peptides and 6 unspiked peptides with a high ("noisy") MSE of 0.882. The heterogeneity across these peptides in the

Table 4.3: Each protein was divided into 2 - 4 clusters based on the reduction in MSE. The number of spiked, unspiked, and invalid peptides is shown for each cluster, along with the MSE calculated on the loading set with sample as the sole predictor. The percentage of unspiked peptides is calculated using the number of valid peptides as the denominator.

Protein	Cluster	Spiked	Unspiked	Invalid	Total	% Unspiked	MSE
HEMO							
	1	70	0	0	70	0.0	0.345
	2	22	6	0	28	21.4	0.882
	Overall	92	6	0	98	6.1	0.543
A1AG1-A1AG2							
	1	27	1	0	28	3.6	0.232
	2	10	2	0	12	16.7	0.533
	3	10	7	1	17	43.8	0.842
	Overall	47	10	1	57	17.9	0.523
APOA1							
	1	44	2	0	46	4.3	0.353
	2	30	4	0	34	11.8	0.536
	3	8	12	0	20	60.0	0.904
	Overall	82	18	0	100	18.0	0.558
APOB							
	1	230	15	0	245	6.1	0.229
	2	41	10	1	51	20.0	0.621
	3	36	2	0	38	5.3	0.575
	4	73	37	0	110	33.6	0.903
	Overall	380	64	1	444	14.4	0.530
CERU							
	1	39	1	0	40	2.5	0.598
	2	58	2	0	60	3.3	0.885
	3	27	0	0	27	0.0	0.824
	4	31	29	1	60	49.2	0.984
	Overall	155	32	1	187	17.2	0.889

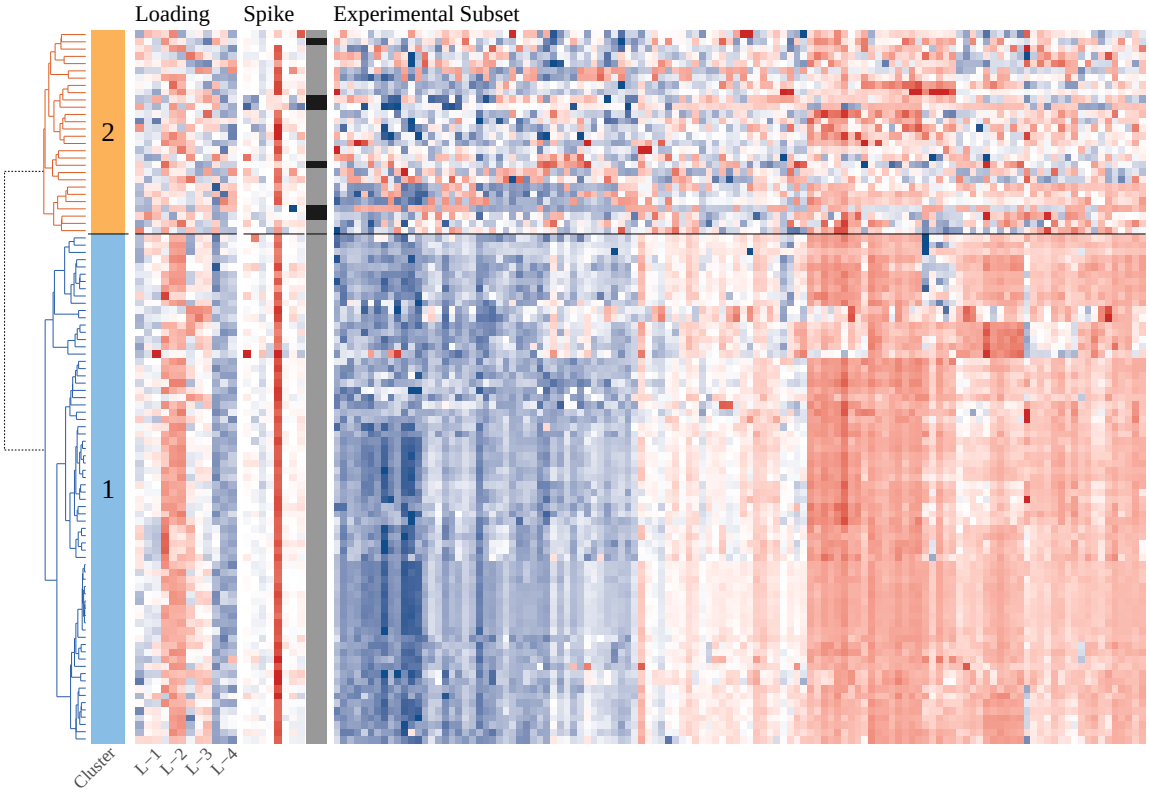


Figure 4.4: Each row of the heat map shows the relative intensity of each peptide across the loading, spike-in, and experimental subsets of the data (columns) for HEMO. Peptides without the spike signal are designated by black.

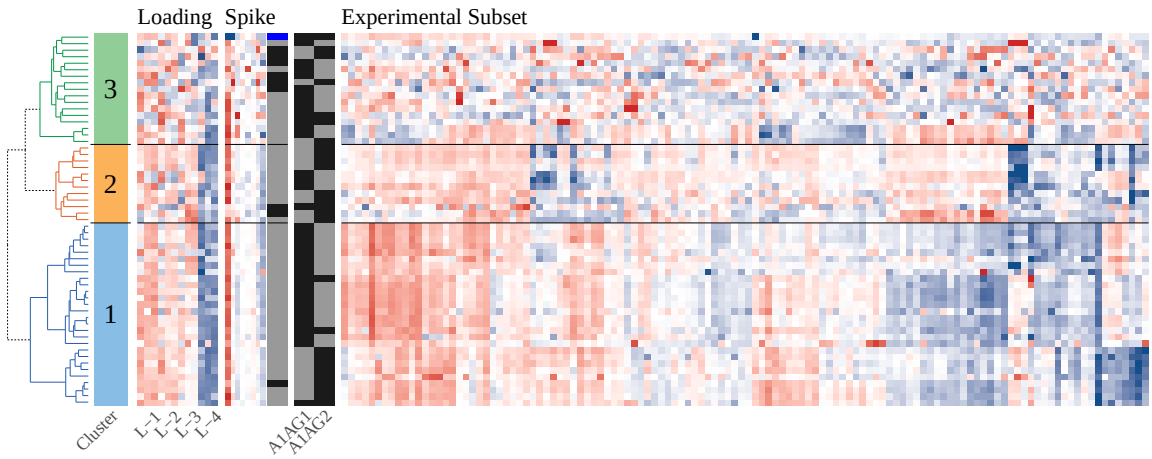


Figure 4.5: Each row of the heat map shows the relative intensity of each peptide across the loading, spike-in, and experimental subsets of the data (columns) for A1AG1 and A1AG2. Peptides without the spike signal are designated by black, while invalid peptides are shown as blue.

experimental subset are correspondingly high and patterns across samples are more difficult to identify.

Peptides belonging to APOA1 (Figure 4.6) and A1AG1-A1AG2 (Fig 4.5) were each split into three clusters: one clean (Cluster 1 with MSEs below 0.40), one noisy (Cluster 3 with MSEs above 0.75), and one intermediate (Cluster 2 with MSEs between 0.40 and 0.75). For both proteins, the Cluster 1 also has the fewest unspiked peptides, with two and one for APOA1 and A1AG1-A1AG2 respectively. Additional annotation is provided for A1AG1-A1AG2 designating whether peptides were mapped to A1AG1, A1AG2, or both. In particular, Cluster 1 for A1AG1-A1AG2 could be divided into two subclusters, with the bottom nine peptides all mapping to A1AG2 and the remaining 18 peptides mapping to A1AG1. These two subclusters have distinct, but correlated abundance patterns across samples. The remaining two clusters of A1AG1-A1AG2 peptides cannot be separated in the same way. Cluster 2 of APOA1 can also be split into subclusters, with all unspiked peptides falling into the top cluster (2 subclusters) and three of four unspiked peptides in a smaller subset of six peptides shown at the very top of Cluster 2. These distinctions illustrate that

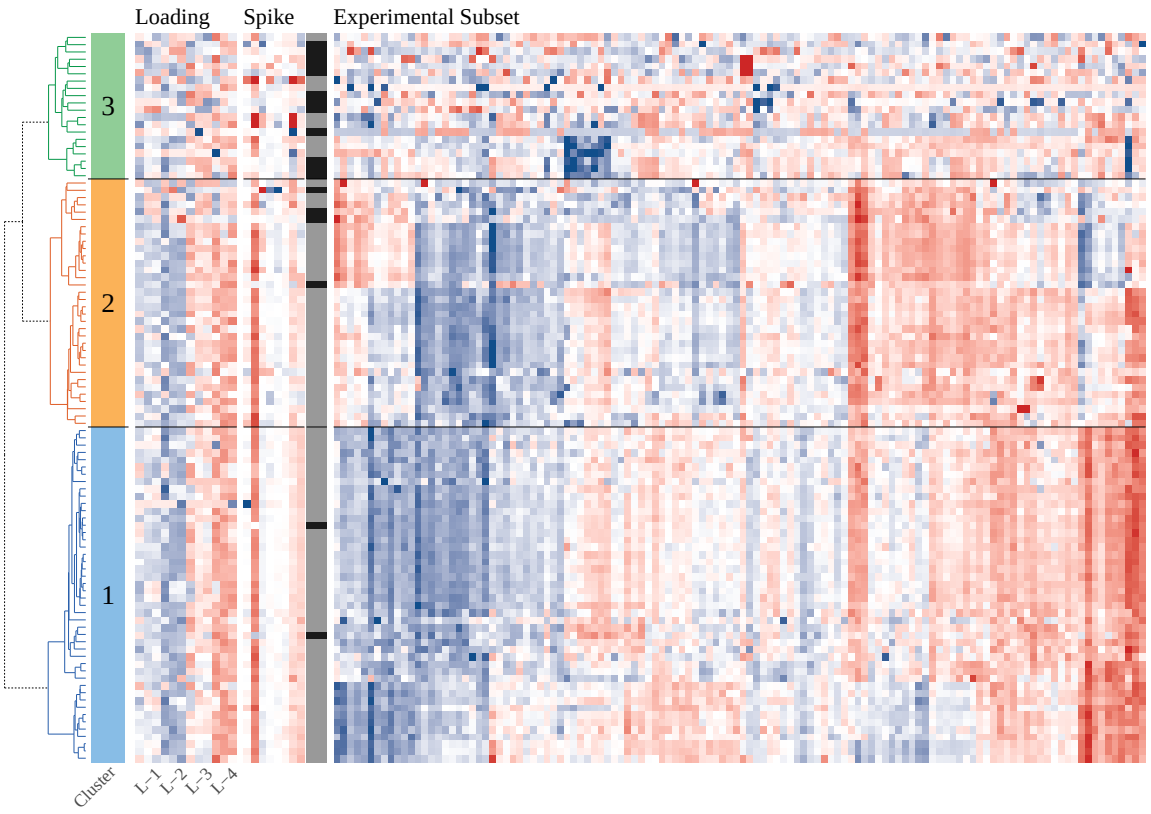


Figure 4.6: Each row of the heat map shows the relative intensity of each peptide across the loading, spike-in, and experimental subsets of the data (columns) for APOA1. Peptides without the spike signal are designated by black.

the selected number of clusters for each protein do not distinguish all patterns in the data, nor are they meant to. For both proteins, the peptides in Cluster 3 contain the highest proportion of unspiked peptides, and have the greatest heterogeneity in both the loading and experimental subsets.

APOB was split into four clusters, with one clean cluster (Cluster 1), two intermediate clusters (Clusters 2 and 3) and one noisy cluster (Cluster 4), as seen in Figure 4.7. As seen in Cluster 2 in the figure, the unspiked peptides in Cluster 2 of APOB are not randomly distributed, but primarily grouped into a subcluster shown at the top of Cluster 2. CERU also was split into four clusters. However, for this protein, the four selected samples for the loading set all had similar intensities, resulting in a high coefficient of variation across mean intensities of each sample. As a result,

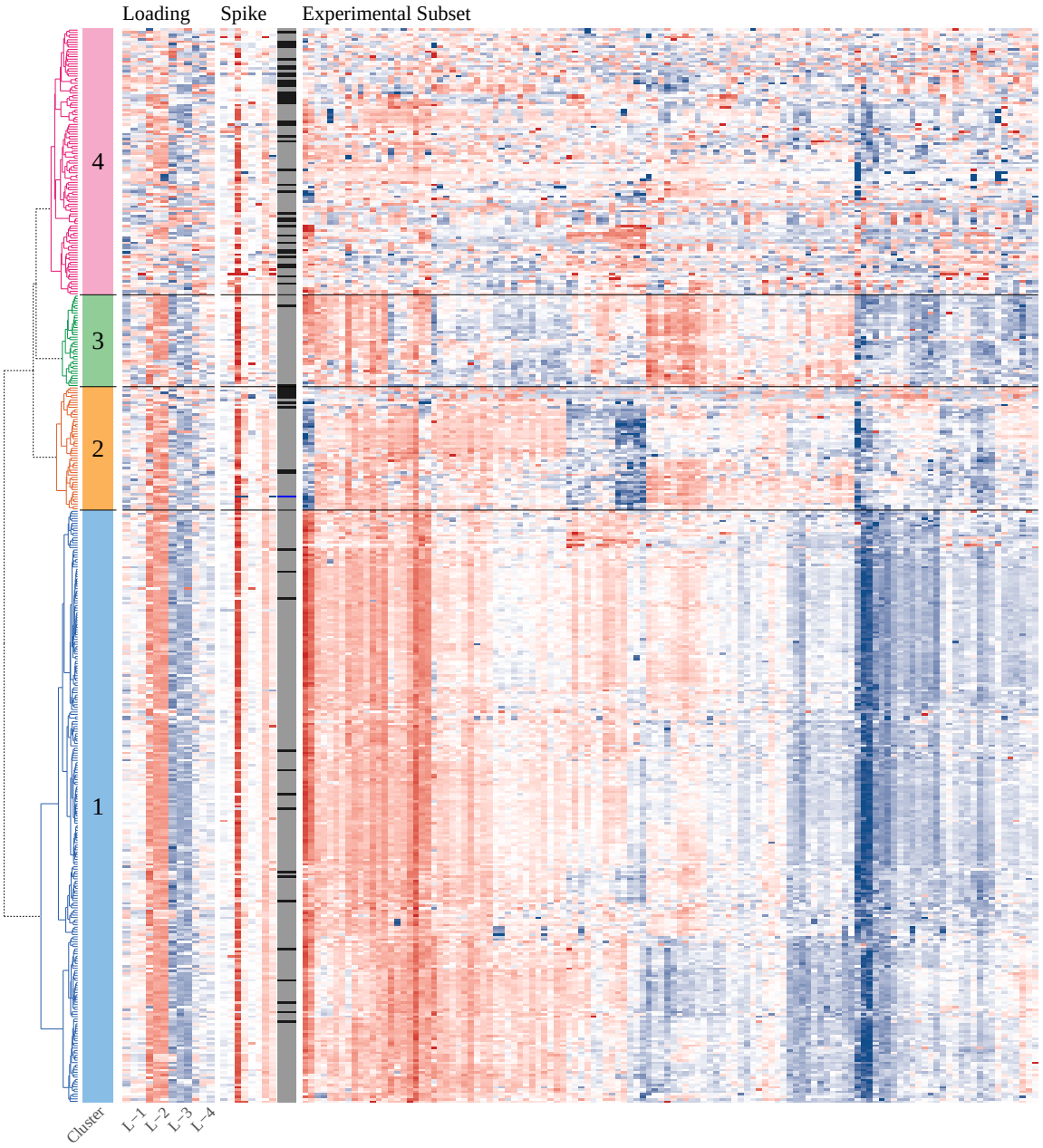


Figure 4.7: Each row of the heat map shows the relative intensity of each peptide across the loading, spike-in, and experimental subsets of the data (columns) for APOB. Peptides without the spike signal are designated by black, while invalid peptides are shown as blue.

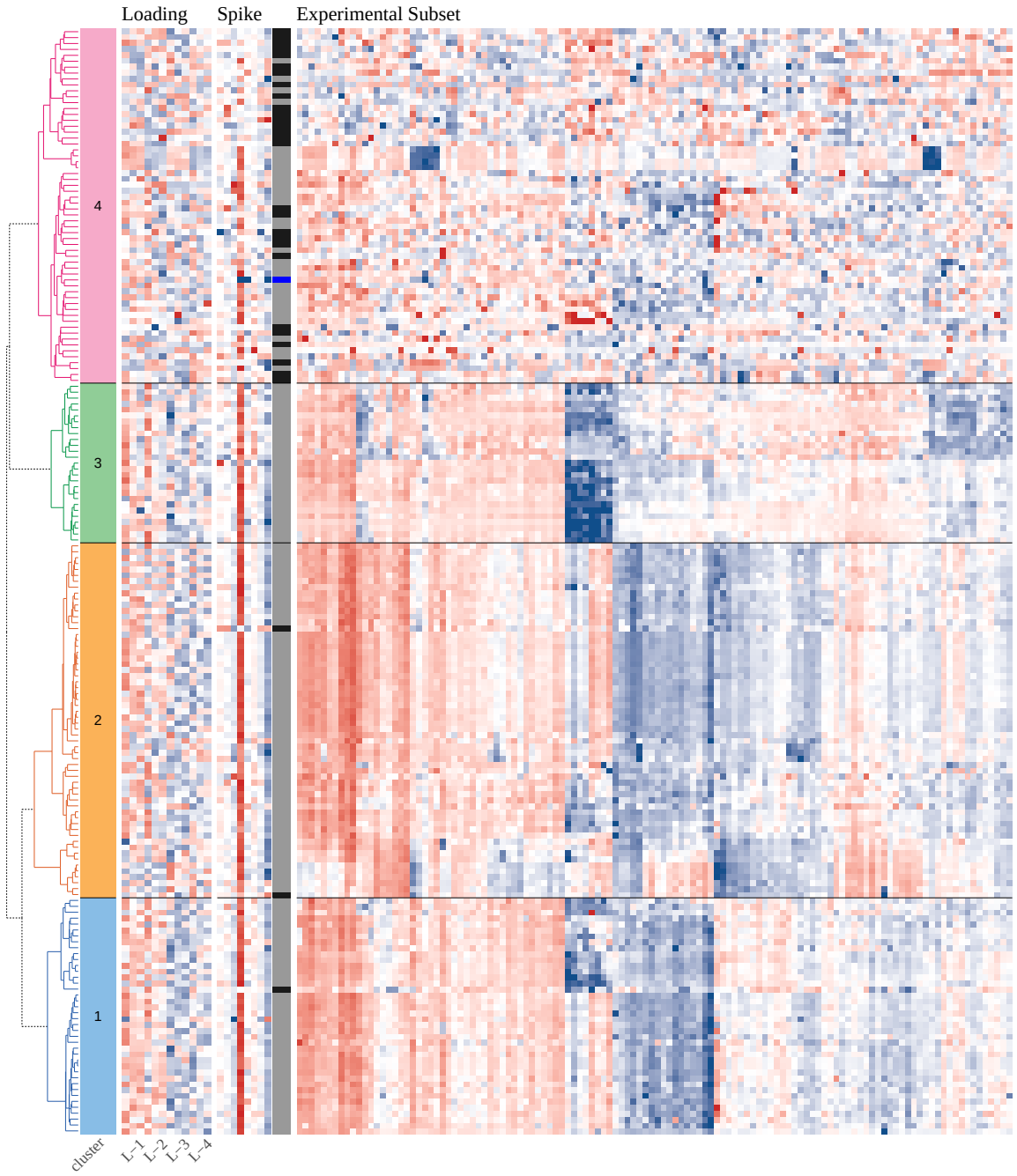


Figure 4.8: Each row of the heat map shows the relative intensity of each peptide across the loading, spike-in, and experimental subsets of the data (columns) for CERU. Peptides without the spike signal are designated by black, while invalid peptides are shown as blue.

Table 4.4: The number of peptides retained and removed from each protein using the three proposed decision rules based on the hierarchical clustering and spike-in subsets.

Protein	**HC**		**Hybrid**		**Spike**	
	Keep	Remove	Keep	Remove	Keep	Remove
HEMO	70	28	70	28	92	6
A1AG1-A1AG2	40	17	38	19	47	10
APOA1	80	20	76	24	82	18
APOB	334	110	322	122	380	64
CERU	127	60	124	63	155	32

the calculated MSE was relatively high for all clusters, with Cluster 1 producing an intermediate MSE and Clusters 2 - 4 producing MSEs classified as noisy. To better reflect the patterns of the experimental data, Clusters 1 - 3 were treated as if they were intermediate, while Cluster 4 was treated as noisy. This decision was supported by the spike-in subset, as 29 out of 32 unspiked proteins were in Cluster 3.

Using these data, three benchmarks were created using the external data, as shown in Table 4.4. The spike decision rule treats the largest number of peptides as correctly identified and is thus the most conservative, with 6.1 % to 18 % of peptides unspiked across the five proteins. Using the HC decision rule, unspiked peptides in clusters with clean or intermediate MSEs were treated as correct, while all peptides in the noisy cluster were treated as incorrect. Universally, the number of spiked peptides in the noisy cluster was larger than the number of unspiked peptides in cleaner clusters, resulting in a net increase in the number of peptides determined to be incorrect relative to the spike benchmark. Between 20 % and 32 % of peptides are noisy using this decision rule. The hybrid rule further increased the number of removed peptides, as unspiked peptides in clusters with intermediate MSEs were removed. Using this criterion, between 24 % and 34 % of each protein is considered noise.

Table 4.5: The estimated FOR, specificity, and sensitivity, and %Δ for `classCleaner` and 14 other algorithms. Since the true status of each peptide is unknown, the results of the spike, HC, and hybrid decision rules are substituted.

		Spike				HC				Hybrid			
Class	Algorithm	FOR	Sens.	Spec.	%Δ	FOR	Sens.	Spec.	%Δ	FOR	Sens.	Spec.	%Δ
HEMO													
Proteomic	classCleaner	0.05	0.33	0.91	-26	0.20	0.36	1.00	-28	0.20	0.36	1.00	-28
	BP-Quant	0.00	1.00	0.76	-100	0.03	0.93	0.97	-90	0.03	0.93	0.97	-90
	PQPQ	0.05	0.17	0.96	-12	0.25	0.18	1.00	-13	0.25	0.18	1.00	-13
adaBoost	EB	0.06	0.00	0.97	3	0.27	0.07	0.99	-4	0.27	0.07	0.99	-4
	ORBoost	0.06	0.00	0.99	1	0.28	0.04	1.00	-3	0.28	0.04	1.00	-3
IBL	BBNR	0.07	0.17	0.77	7	0.36	0.04	0.70	24	0.36	0.04	0.70	24
	ENG	0.01	0.83	0.78	-78	0.08	0.79	0.96	-71	0.08	0.79	0.96	-71
	ENN	0.01	0.83	0.73	-76	0.04	0.89	0.93	-85	0.04	0.89	0.93	-85
	GE	0.03	0.67	0.74	-53	0.09	0.79	0.91	-70	0.09	0.79	0.91	-70
Voting	V-C4.5	0.02	0.67	0.90	-62	0.18	0.46	1.00	-38	0.18	0.46	1.00	-38
	CVCF	0.02	0.67	0.97	-64	0.23	0.25	1.00	-19	0.23	0.25	1.00	-19
	EF	0.03	0.67	0.78	-56	0.08	0.79	0.97	-72	0.08	0.79	0.97	-72
	HARF	0.02	0.83	0.62	-72	0.02	0.96	0.81	-94	0.02	0.96	0.81	-94
	PF	0.06	0.00	1.00	0	0.29	0.00	1.00	0	0.29	0.00	1.00	0
Iterative	R-C4.5	0.00	1.00	0.61	-100	0.05	0.89	0.76	-81	0.05	0.89	0.76	-81
	I-C4.5	0.00	1.00	0.71	-100	0.05	0.89	0.89	-84	0.05	0.89	0.89	-84
No Filter		0.06	0.00	1.00	0	0.29	0.00	1.00	0	0.29	0.00	1.00	0

(continued)

		Spike				HC				Hybrid			
Class	Algorithm	FOR	Sens.	Spec.	%Δ	FOR	Sens.	Spec.	%Δ	FOR	Sens.	Spec.	%Δ
A1AG1-A1AG2													
Proteomic	classCleaner	0.11	0.5	0.87	-38	0.17	0.53	0.95	-42	0.20	0.53	0.97	-41
	BP-Quant	0.00	1.0	0.30	-100	0.00	1.00	0.35	-100	0.00	1.00	0.37	-100
	PQPQ	0.14	0.4	0.79	-20	0.19	0.53	0.88	-38	0.21	0.53	0.89	-37
adaBoost	EB	0.15	0.3	0.83	-13	0.26	0.29	0.85	-13	0.26	0.37	0.89	-22
	ORBoost	0.18	0.0	1.00	0	0.30	0.00	1.00	0	0.33	0.00	1.00	0
IBL	BBNR	0.15	0.3	0.85	-15	0.28	0.24	0.85	-7	0.30	0.26	0.87	-11
	ENG	0.04	0.9	0.53	-78	0.08	0.88	0.60	-74	0.08	0.89	0.63	-77
	ENN	0.06	0.9	0.32	-64	0.12	0.88	0.35	-58	0.12	0.89	0.37	-62
	GE	0.20	0.8	0.17	14	0.10	0.94	0.22	-66	0.10	0.95	0.24	-70
Voting	V-C4.5	0.18	0.0	0.98	2	0.29	0.06	1.00	-4	0.32	0.05	1.00	-4
	CVCF	0.18	0.0	0.98	2	0.29	0.06	1.00	-4	0.32	0.05	1.00	-4
	EF	0.07	0.7	0.83	-59	0.10	0.76	0.95	-68	0.12	0.74	0.97	-64
	HARF	0.00	1.0	0.13	-100	0.00	1.00	0.15	-100	0.00	1.00	0.16	-100
	PF	0.18	0.0	1.00	0	0.30	0.00	1.00	0	0.33	0.00	1.00	0
Iterative	R-C4.5	0.23	0.7	0.21	32	0.00	1.00	0.32	-100	0.15	0.89	0.29	-54
	I-C4.5	0.06	0.9	0.34	-66	0.06	0.94	0.40	-80	0.06	0.95	0.42	-82
No Filter		0.18	0.0	1.00	0	0.30	0.00	1.00	0	0.33	0.00	1.00	0

(continued)

		Spike				HC				Hybrid			
Class	Algorithm	FOR	Sens.	Spec.	%Δ	FOR	Sens.	Spec.	%Δ	FOR	Sens.	Spec.	%Δ
APOA1													
Proteomic	classCleaner	0.07	0.67	0.96	-61	0.09	0.60	0.96	-53	0.12	0.58	0.99	-51
	BP-Quant	0.04	0.89	0.59	-78	0.00	1.00	0.62	-100	0.00	1.00	0.66	-100
	PQPQ	0.07	0.67	0.96	-61	0.11	0.55	0.95	-47	0.12	0.58	0.99	-51
adaBoost	EB	0.18	0.11	0.91	-2	0.20	0.10	0.91	-1	0.23	0.12	0.92	-4
	ORBoost	0.21	0.00	0.82	18	0.22	0.05	0.82	12	0.27	0.04	0.82	13
IBL	BBNR	0.23	0.33	0.49	28	0.25	0.35	0.49	25	0.29	0.38	0.49	20
	ENG	0.04	0.89	0.61	-79	0.02	0.95	0.64	-90	0.02	0.96	0.67	-92
	ENN	0.05	0.89	0.50	-74	0.02	0.95	0.52	-88	0.02	0.96	0.55	-90
	GE	0.10	0.78	0.45	-46	0.10	0.80	0.46	-51	0.10	0.83	0.49	-59
Voting	V-C4.5	0.16	0.11	1.00	-9	0.18	0.10	1.00	-8	0.22	0.08	1.00	-6
	CVCF	0.16	0.11	1.00	-9	0.18	0.10	1.00	-8	0.22	0.08	1.00	-6
	EF	0.05	0.78	0.96	-73	0.08	0.65	0.95	-58	0.10	0.67	0.99	-60
	HARF	0.05	0.94	0.22	-71	0.05	0.95	0.22	-74	0.05	0.96	0.24	-78
	PF	0.18	0.00	1.00	0	0.20	0.00	1.00	0	0.24	0.00	1.00	0
Iterative	R-C4.5	0.09	0.83	0.39	-52	0.03	0.95	0.42	-86	0.06	0.92	0.43	-76
	I-C4.5	0.03	0.94	0.44	-85	0.05	0.90	0.44	-73	0.05	0.92	0.46	-77
No Filter		0.18	0.00	1.00	0	0.20	0.00	1.00	0	0.24	0.00	1.00	0

(continued)

		Spike				HC				Hybrid			
Class	Algorithm	FOR	Sens.	Spec.	%Δ	FOR	Sens.	Spec.	%Δ	FOR	Sens.	Spec.	%Δ
APOB													
Proteomic	classCleaner	0.09	0.53	0.84	-40	0.13	0.59	0.91	-48	0.14	0.59	0.93	-48
	BP-Quant	0.08	0.70	0.58	-45	0.07	0.85	0.67	-71	0.08	0.84	0.68	-70
	PQPQ	0.07	0.67	0.69	-49	0.16	0.57	0.71	-33	0.17	0.60	0.73	-37
adaBoost	EB	0.14	0.05	0.98	-2	0.24	0.07	0.99	-5	0.26	0.07	0.99	-4
	ORBoost	0.14	0.00	1.00	0	0.25	0.00	1.00	0	0.27	0.00	1.00	0
IBL	BBNR	0.12	0.36	0.76	-13	0.18	0.47	0.81	-29	0.20	0.45	0.81	-26
	ENG	0.08	0.55	0.84	-43	0.10	0.67	0.94	-58	0.12	0.66	0.96	-56
	ENN	0.08	0.56	0.84	-44	0.10	0.70	0.94	-62	0.11	0.68	0.95	-59
	GE	0.11	0.41	0.85	-27	0.13	0.57	0.94	-47	0.15	0.57	0.95	-46
Voting	V-C4.5	0.14	0.08	0.99	-6	0.24	0.05	1.00	-4	0.26	0.06	1.00	-4
	CVCF	0.14	0.06	0.99	-5	0.24	0.05	1.00	-4	0.26	0.05	1.00	-4
	EF	0.08	0.52	0.90	-42	0.13	0.57	0.97	-49	0.14	0.57	0.99	-48
	HARF	0.07	0.66	0.74	-50	0.07	0.82	0.85	-73	0.08	0.80	0.87	-71
	PF	0.14	0.00	1.00	0	0.25	0.00	1.00	0	0.27	0.00	1.00	0
Iterative	R-C4.5	0.09	0.55	0.80	-40	0.13	0.61	0.87	-48	0.14	0.61	0.89	-48
	I-C4.5	0.10	0.47	0.82	-32	0.12	0.63	0.92	-52	0.14	0.61	0.93	-50
No Filter		0.14	0.00	1.00	0	0.25	0.00	1.00	0	0.27	0.00	1.00	0

(continued)

		Spike				HC				Hybrid			
Class	Algorithm	FOR	Sens.	Spec.	%Δ	FOR	Sens.	Spec.	%Δ	FOR	Sens.	Spec.	%Δ
CERU													
Proteomic	classCleaner	0.09	0.59	0.90	-50	0.23	0.42	0.92	-28	0.24	0.41	0.93	-28
	BP-Quant	0.03	0.91	0.74	-85	0.11	0.78	0.82	-65	0.11	0.79	0.84	-67
	PQPQ	0.10	0.47	0.94	-39	0.25	0.33	0.97	-24	0.26	0.32	0.97	-22
adaBoost	EB	0.15	0.19	0.95	-12	0.28	0.18	0.98	-12	0.30	0.17	0.98	-11
	ORBoost	0.17	0.00	1.00	0	0.32	0.00	1.00	0	0.34	0.00	1.00	0
IBL	BBNR	0.15	0.44	0.66	-13	0.31	0.37	0.65	-2	0.32	0.38	0.66	-4
	ENG	0.04	0.84	0.73	-75	0.10	0.80	0.83	-68	0.11	0.79	0.85	-67
	ENN	0.04	0.88	0.66	-78	0.10	0.82	0.76	-68	0.11	0.81	0.77	-67
	GE	0.12	0.56	0.64	-28	0.17	0.68	0.74	-48	0.19	0.67	0.74	-45
Voting	V-C4.5	0.15	0.19	0.99	-15	0.29	0.13	1.00	-9	0.31	0.13	1.00	-9
	CVCF	0.15	0.19	0.99	-15	0.30	0.12	0.99	-8	0.31	0.13	1.00	-9
	EF	0.05	0.78	0.90	-72	0.15	0.63	0.98	-53	0.16	0.62	0.98	-51
	HARF	0.00	1.00	0.43	-100	0.00	1.00	0.52	-100	0.00	1.00	0.53	-100
	PF	0.17	0.00	1.00	0	0.32	0.00	1.00	0	0.34	0.00	1.00	0
Iterative	R-C4.5	0.06	0.81	0.56	-62	0.15	0.77	0.62	-53	0.16	0.76	0.63	-52
	I-C4.5	0.02	0.94	0.66	-89	0.10	0.82	0.74	-67	0.10	0.83	0.76	-69
No Filter		0.17	0.00	1.00	0	0.32	0.00	1.00	0	0.34	0.00	1.00	0

4.3.2 ClassCleaner **results**

Table 4.5 compares the results of each algorithm with three benchmarks, based on spike, HC, and hybrid decision rules. For consistency with previous assessments, the FOR, specificity, sensitivity, and %Δ were calculated under the assumption that each benchmark is correct. This provides a consistent, albeit imperfect, set of references against which to judge each algorithm.

ClassCleaner removed ten peptides from HEMO. All of these peptides were in Cluster 2, and they included two of the six unspiked peptides, resulting in a 25.8 % or 28.4 % reduction in the estimated FOR depending on whether the spiked or HC benchmark was used to estimate the truth. (The hybrid rule produces identical results to the HC rule for this protein.) A closer look at the experimental data set, coupled with the knowledge that only six out of 28 of these peptides were unspiked, suggests that many of the peptides in Cluster 2 are correctly identified, but have a low signal-to-noise ratio. In general, noisy peptides are expected to have smaller correlations relative to those measured accurately. For this protein, these correlations were large enough to produce over $a_{0.05/98} = 65$ distances below t^*, resulting in many of these peptides being retained by classCleaner.

When applied to A1AG1-A1AG2, classCleaner removed two peptides from Cluster 2 (one unspiked) and nine peptides from Cluster 3 (four unspiked), as shown in Figure 4.10. In total, five of the eight unspiked peptides were removed, resulting in a -%Δ between 38 % and 42 %, depending on which benchmark is used for comparison. Of the five spiked peptides in Cluster 3 with values below the cut-off of 24, three formed a highly correlated subset shown at the very bottom of Cluster 3. These peptides, while highly correlated with each other, were not correlated with many of the other peptides in the protein, and thus removed. The specificity across the three benchmarks was between 0.87 (spike) and 0.97 (hybrid).

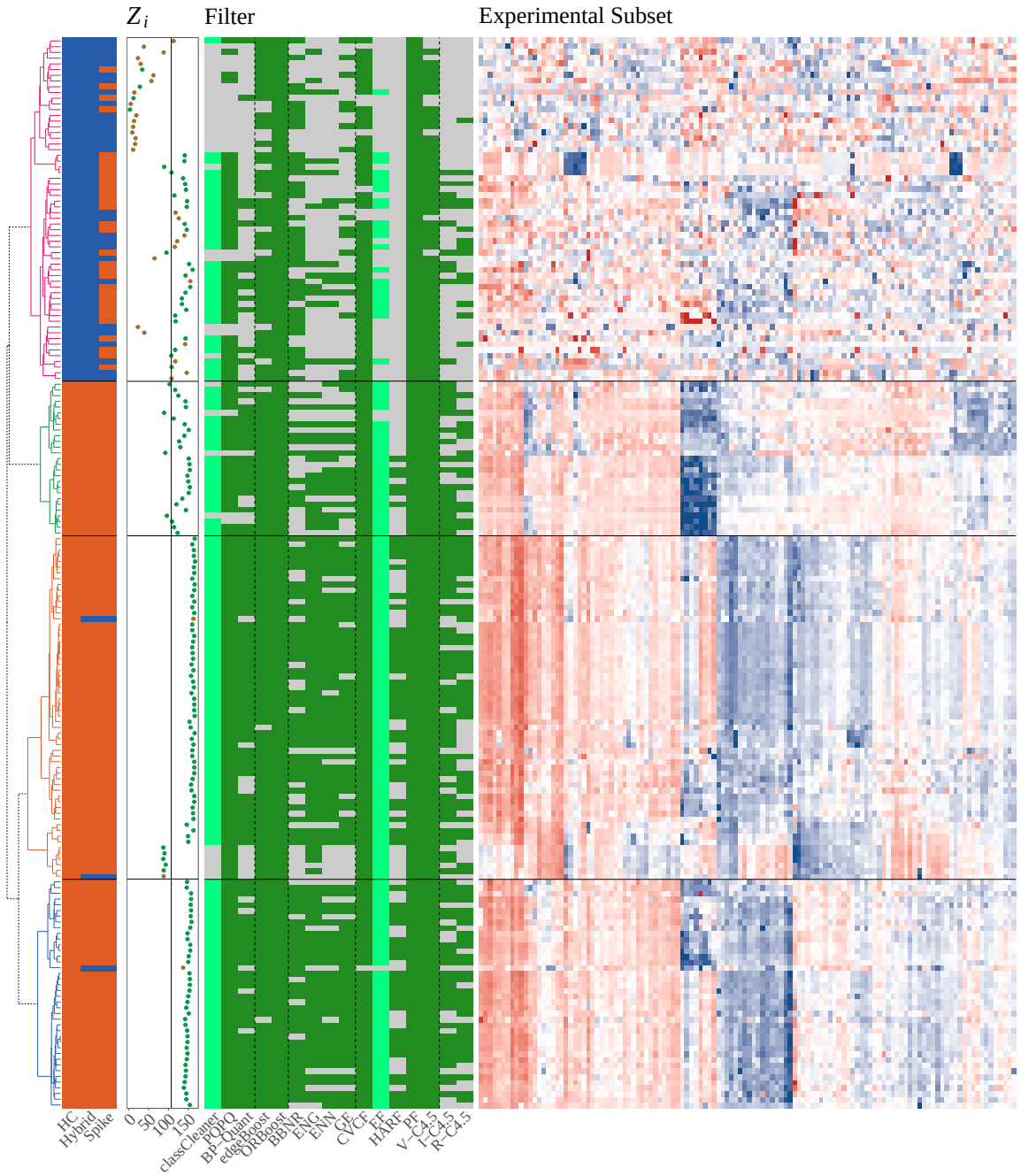


Figure 4.9: Each row (peptide) of the experimental data set on HEMO is annotated with (1) whether the peptide was good (orange) or bad (blue) based on the three benchmarks; (2) The value of Z_i from the classCleaner algorithm; and (3) whether each filtering algorithm retained (green) or removed (gray) the peptide. The results of classCleaner and EF are highlighted for easier comparison.

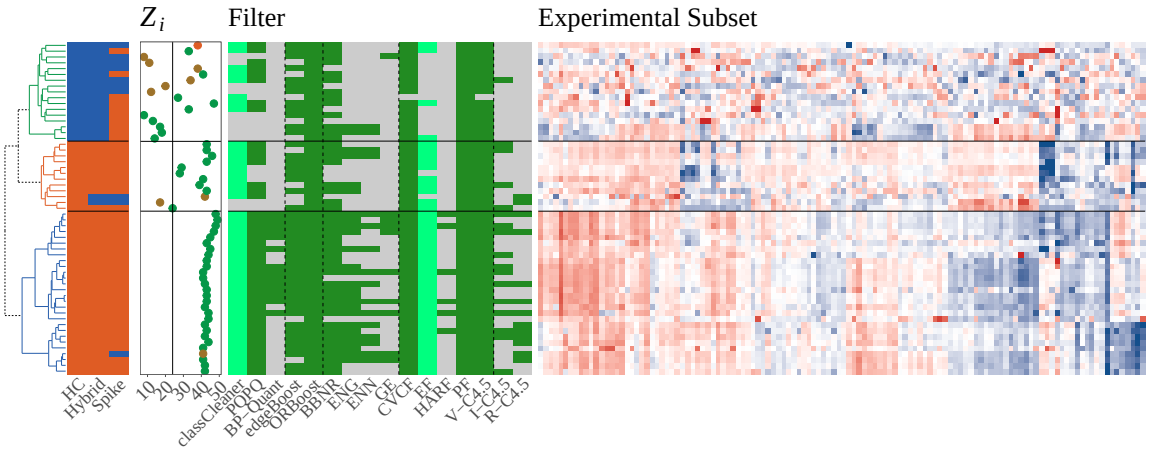


Figure 4.10: Each row (peptide) of the experimental data set on A1AG1-A1AG2 is annotated with (1) whether the peptide was considered good (orange) or bad (blue) based on the three benchmarks; (2) The value of Z_i from the classCleaner algorithm; and (3) whether each filtering algorithm retained (green) or removed (gray) the peptide. The results of classCleaner and EF are highlighted for easier comparison.

classCleaner removed 15 peptides from APOA1, among them 12 of the 18 unspiked peptides, as shown in Figure 4.11. This produced a decrease in the estimated FOR between 51 % and 60.8 % and an estimated specificity between 0.96 and 0.99, depending on which benchmark was used.

ClassCleaner removed 96 total peptides from APOB, including 19 in Cluster 2, 12 in Cluster 3, and 65 in Cluster 4. Thirty-four (34) were among the 64 unspiked peptides, resulting in a percent decrease in the estimated FOR between 40.2 % and 47.8 %, and an estimated specificity between 0.84 and 0.93. Of particular interest among these results is a group of peptides in Cluster 4 with high values of Z_i, as seen in Figure 4.12. The experimental data for these proteins suggests that while they are noisier than peptides included in Clusters 1 and 2, they show the same pattern of high intensities in samples displayed on the left (red) and low intensities in samples displayed on the right (blue). Other algorithms, including PQPQ, R-C4.5, I-C4.5, HARF, EF, ENN, ENG, and GE, also preferentially retained these peptides. While the true status of these peptides is unknown, if the consensus behavior is correct

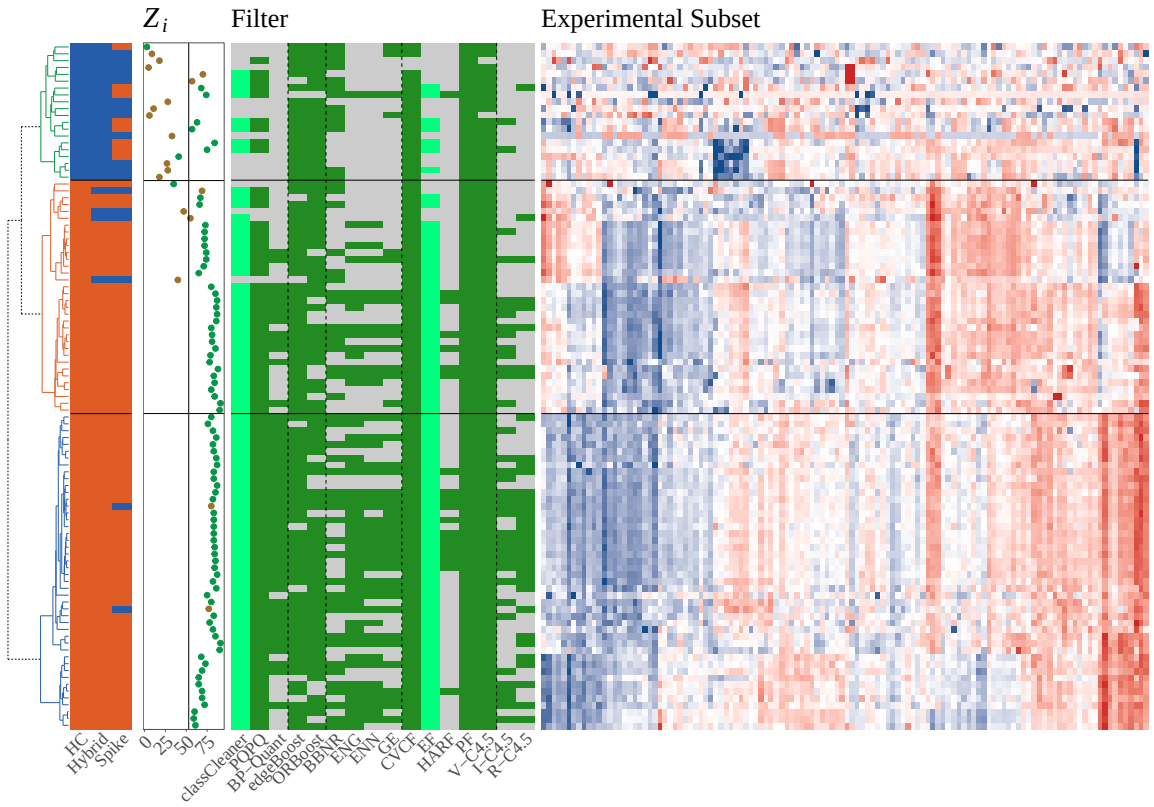


Figure 4.11: Each row (peptide) of the experimental data set on APOA1 is annotated with (1) whether the peptide was considered good (orange) or bad (blue) based on the three benchmarks; (2) The value of Z_i from the classCleaner algorithm; and (3) whether each filtering algorithm retained (green) or removed (gray) the peptide. The results of classCleaner and EF are highlighted for easier comparison.

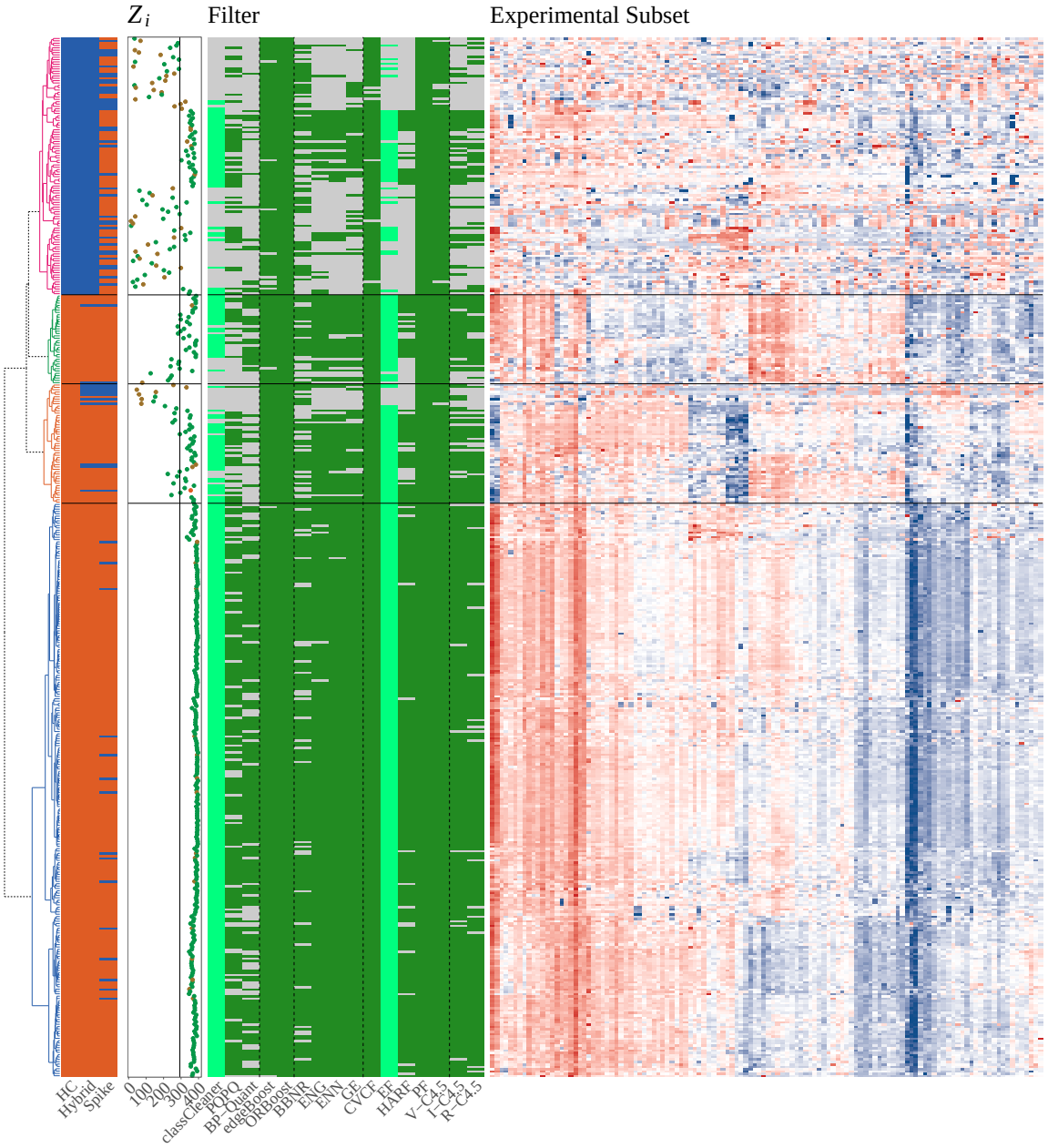


Figure 4.12: In this heat map of APOB intensities, each row (peptide) of the experimental data set is annotated with (1) whether the peptide was considered good (orange) or bad (blue) based on the three benchmarks; (2) The value of Z_i from the classCleaner algorithm, with the line showing the value of a_α for the protein; and (3) whether each filtering algorithm retained (green) or removed (gray) the peptide. The results of classCleaner and EF are highlighted for easier comparison.

then the FOR is likely overestimated and specificity underestimated when the HC or hybrid benchmarks are used.

ClassCleaner removed 35 total peptides from CERU, including six in Cluster 2, four in Cluster 3. Nineteen (19) were among the 32 unspiked peptides, resulting in a percent decrease in the estimated FOR between 27.7 % and 50.0 %, and an estimated specificity between 0.90 and 0.93. The large range in the percent decrease in FOR is a consequence of the subclusters within Cluster 4, as illustrated in Figure 4.13. Three subclusters of Cluster 4 can be distinguished: 36 peptides at the bottom (12 were unspiked and 1 was invalid) appear to be noisy versions of Cluster 1; 4 spiked peptides with very high correlations to one another in the middle, and 20 peptides at the top (16 were unspiked) with no pattern across the samples. classCleaner removes 19 peptides in the top cluster, one from the middle cluster, and five from the bottom cluster. This suggests that most of the peptides across the bottom were sufficiently correlated with other peptides to ensure that over 107 distances were below the distance threshold despite poor signal-to-noise ratios. In contrast, peptides in the top subcluster were almost universally removed.

Figure 4.14 summarize the results of classCleaner using each benchmark across the five proteins. While the FOR was highly dependent upon the selected benchmark, using $\%\Delta$ provided far more stable results, showing a 30 % to 70 % decrease in the proportion of undesired peptides in the final data set. The specificity was lower using the spike benchmark because only peptides presumably mislabeled – not those with a low signal-to-noise ratio – are treated as incorrect using this benchmark. Using the HC and hybrid benchmarks, the specificity of the algorithm was over 90 % for all five proteins.

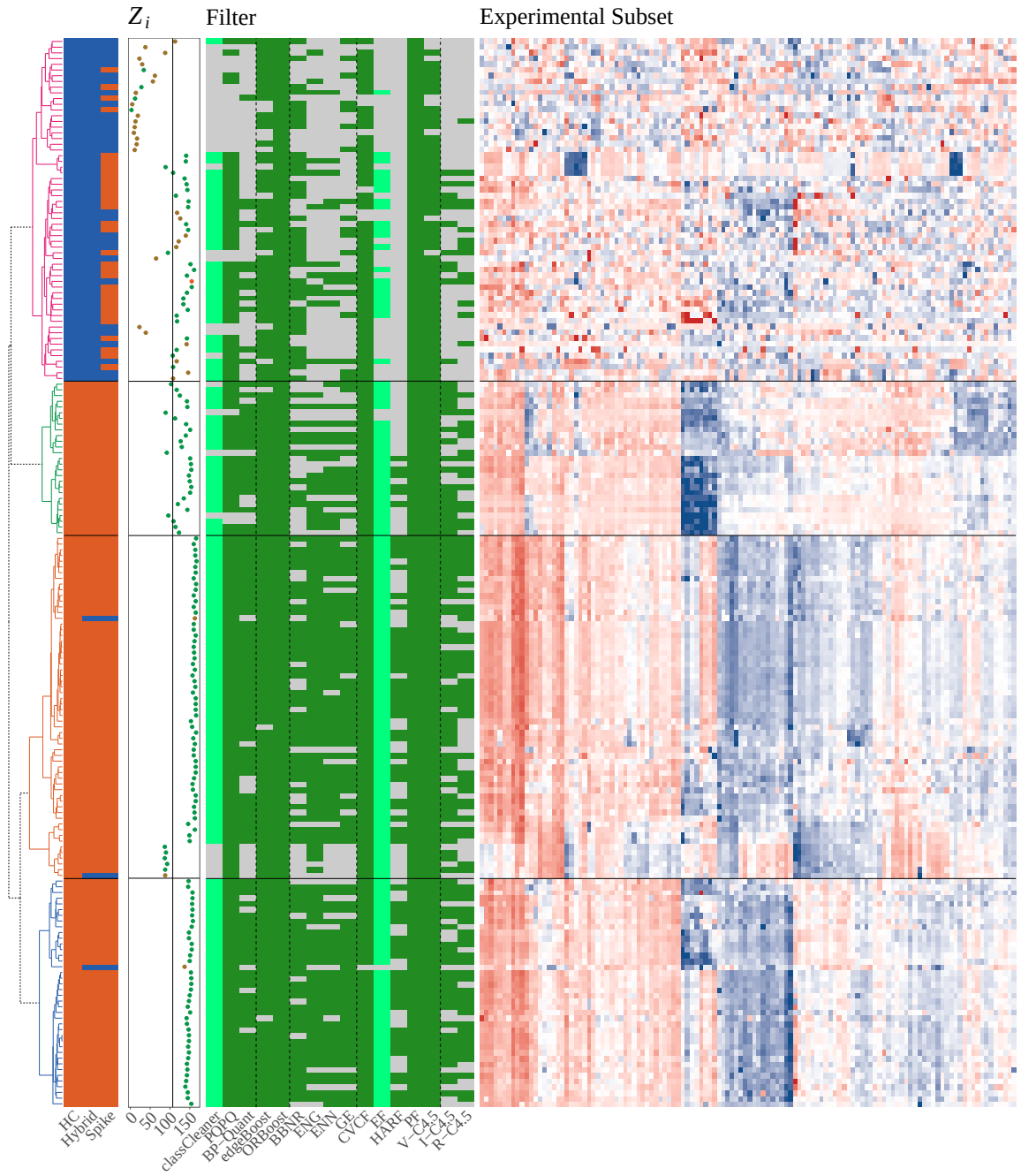


Figure 4.13: In this heat map of CERU intensities, each row (peptide) of the experimental data set is annotated with (1) whether the peptide was considered good (orange) or bad (blue) based on the three benchmarks; (2) The value of Z_i from the classCleaner algorithm, with the line showing the value of a_α for the protein; and (3) whether each filtering algorithm retained (green) or removed (gray) the peptide. The results of classCleaner and EF are highlighted for easier comparison.

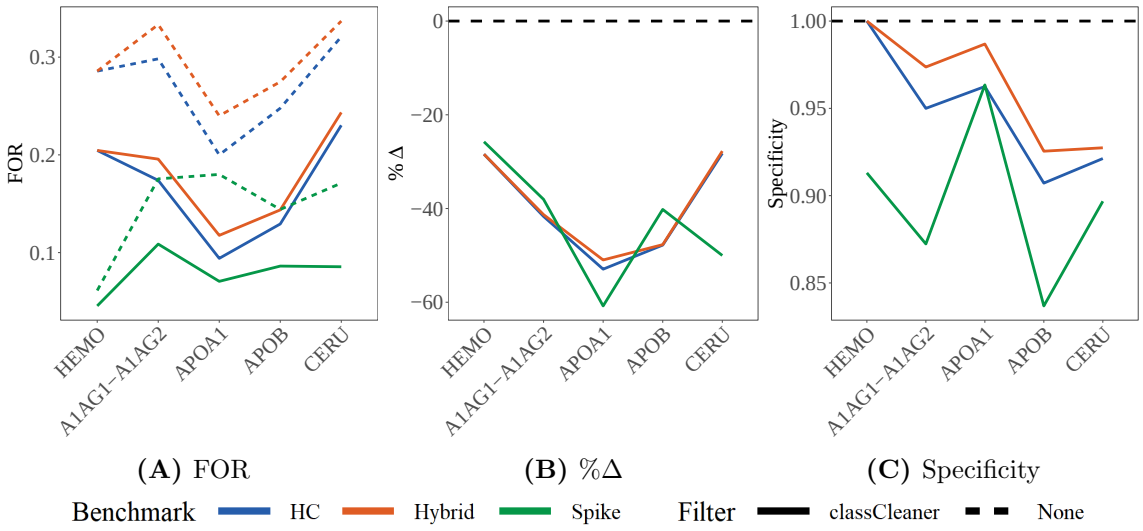


Figure 4.14: Estimated FOR, specificity, and %Δ from classCleaner over all five proteins.

4.3.3 Comparing classCleaner to other algorithms

Table 4.5 also shows the estimated sensitivity, specificity, FOR, and %Δ for fourteen comparative algorithm. To better illustrate the general trends across these algorithms, they were grouped into categories based on the specificity and %Δ, as seen in Figure 4.15. Many algorithms had behavior dependent upon the size and initial misidentification rate, as shown by the difference in results between A1AG1-A1AG2 and APOA1 (groups surrounded by solid lines) versus the APOB, CERU, and HEMO (dashed lines). These trends are most visible when the algorithms are compared to the hybrid or HC benchmarks.

ClassCleaner, PQPQ, and EF formed the first group, show in in blue. These algorithms were characterized by consistent results across all five proteins including a high specificity and a moderate decrease in %Δ across all five proteins. This was also the only group where the results from A1AG1-A1AG2 and APOA1 had a high degree of overlap with the other three proteins. The best performance was observed by EF, which had a specificity above 95 % using the HC and hybrid benchmark, and between 0.78 to 0.97 using the spike benchmark. The FOR decreased by 42 % to

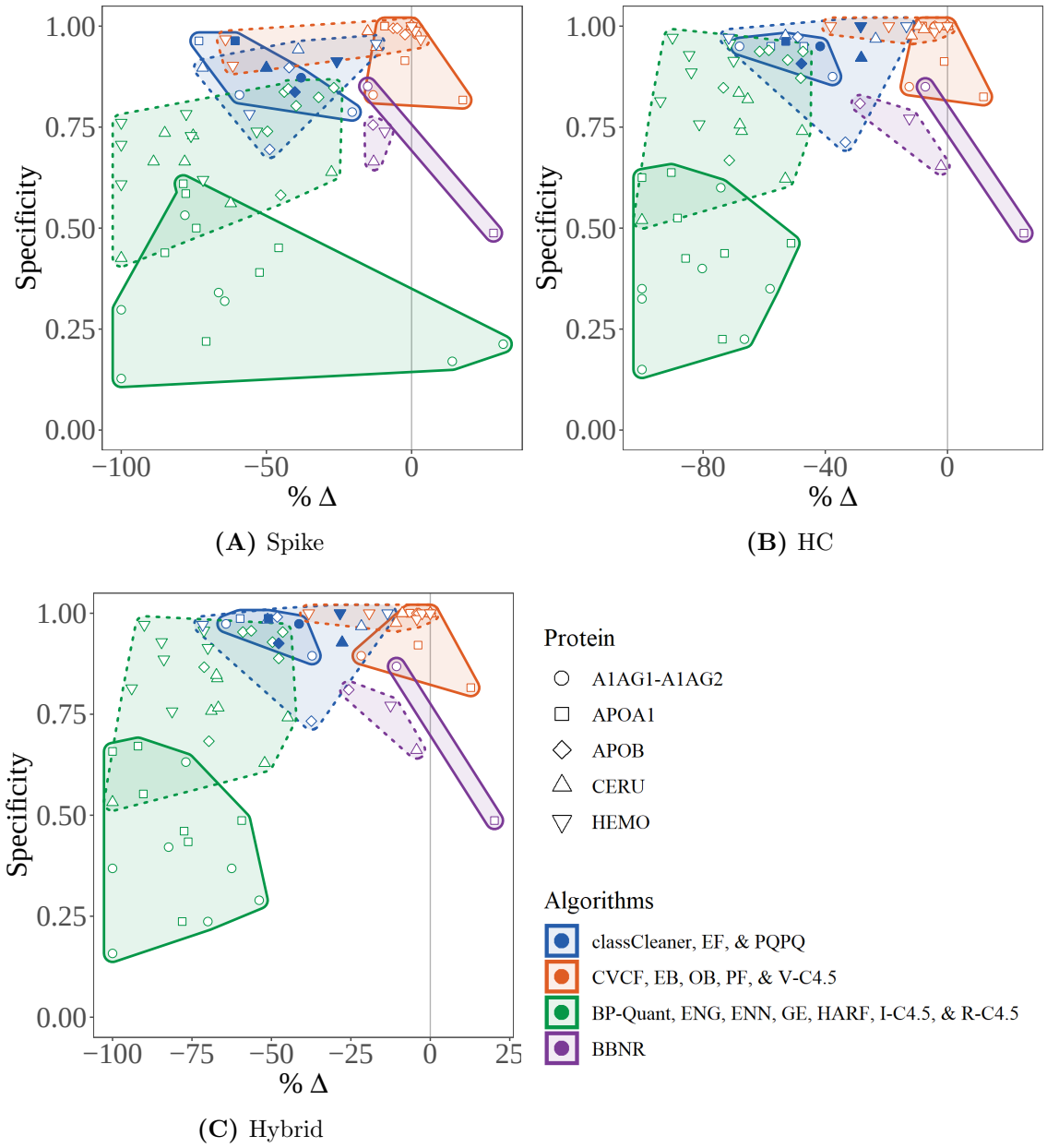


Figure 4.15: The results of each algorithm, grouped by the outcome on A1AG1-A1AG2 and APOA1 (solid line) and APOB, CERU, and HEMO (dashed line). The filled points show the results from classCleaner.

73.2 %, depending on the protein and the decision rule. classCleaner had a similar performance, with a specificity above 0.90 using the HC and hybrid benchmark, and between 0.83 to 0.97 using the spike benchmark. The FOR decreased by 25.8 % to 60.8 %. Excluding APOB, the minimum specificity of PQPQ was 0.787 using the spike benchmark and at least 0.875 using the remaining benchmark. The specificity for APOB was between 0.695 to 0.733 due to changes in how the algorithm behaves for proteins with over 300 peptides. This change did not appear to affect the FOR, where -%Δ was between 12.2 % and 60.8 %.

The second group of algorithms, shown in orange in Figure 4.15, were characterized by the high proportion of retained peptides. CVCF, EB, ORBoost, PF, and V-C4.5 kept 94 % to 100 % of peptides across the five protein. These high retention rates produced a high specificity, but resulted in almost no reduction in the FOR, especially for A1AG1-A1AG2 and APOA1 (the darker shaded region in the figure). The most extreme case was PF, which did not remove a single peptide across all proteins. ORBoost removed 16 peptides across all five proteins. Of these, 14 (all from APOA1)were judged correct by all three benchmarks, for a net increase in the proportion of misidentified peptides in this protein. In hindsight, this was determined to be the result of an undocumented issue with the implementation of the algorithm in multi-class problems.

BP-Quant, ENG, R-C4.5, I-C4.5, HARF, GE, and ENN were characterized by a low proportion of retained peptides, with only 51 % to 70 % of peptides retained over all five proteins. The retention rates for these algorithms was highly dependent on the proteins, as exhibited by the difference in the specificity across the two groups of proteins shown in green in Figure 4.15. For example, GE kept only 17.5 % of peptides from A1AG1-A1AG2, but 81.1 % from APOB. In almost all cases, these algorithms did preferentially remove incorrect peptides, with -%Δ above 50 % for all algorithms when the HC or hybrid benchmark was used, and for all but two algorithms when the

spike benchmark was used. GE and R-C4.5 both cause an increase in the %Δ when using the spike benchmark. This appears to be a result of their respectively poor retention rates on A1AG1-A1AG2: so few peptides were retained by the algorithms that the presence of one or two incorrect peptides was sufficient to increase the FOR.

The low retention rate for BP-Quant can be primarily attributed to the algorithm removing all peptides without a significant p-value for any contrast, and is the primary reason for a low specificity in A1AG1-A1AG2 and APOB. Furthermore, because BP-Quant uses the results from test statistics instead of the raw data, it was very sensitive to small changes in peptide abundance. For example, several peptides in APOB were found to have lower-than-expected intensity measurements for several samples. As a result of these outliers, the peptides no longer had a significant contrast, and were not retained by the algorithm despite a general high correlation with significant peptides. In addition, the algorithm missed some peptides because it could not associate similar significance profiles. Among the peptides of CERU, BP-Quant retained 117 peptides with significance profiles {0, 0, 1, 0, 0, 0} and {1, 1, 1, 0, 0, 0}, but removed nine additional peptides with significance profile {0, 1, 1, 0, 0, 0}. BP-Quant does successfully remove many unspiked and noisy peptides. However, it is biased towards significant peptides, and can miss peptides which differ from accepted peptides only slightly when the p-value is near the significance threshold.

BBNR retained 70 % of the peptides, with a specificity between 0.48 and 0.87 across all five proteins and benchmarks. As shown in Figure 4.15, this is lower than the specificity observed for CVCF, EB, ORBoost, PF, and V-C4.5 while similar to that of BP-Quant, ENG, ENN, GE, HARF, I-C4.5, and R-C4.5. The FOR (−15.1 % to 28.2 % using the spike benchmark, −28.1 % to 25 % using the HC benchmark, and −25.7 % to 20.2 % using the hybrid benchmark) was the reverse: similar to CVCF, EB, ORBoost, PF, and V-C4.5 and higher than BP-Quant, ENG, ENN, GE, HARF, I-C4.5, and R-C4.5. This unusual behavior is likely due to the difference in approach

this algorithm takes. While most of the classification filtering algorithms attempt to improve classification by removing instances likely to be mislabeled, BBNR removes instances which cause other instances to be misclassified. Because proteomics data sets tend to have overlapping clusters due to inherent correlations between proteins, it is very likely that the observed behavior can be attributed to correctly identified peptides from proteins that are highly correlated with other proteins. BBNR is far more likely to remove such peptides, as their nearest neighbors may originate from these correlated proteins.

Chapter 5

An analysis of congressional voting records in 1984

5.1 Introduction

The Congressional Voting Records Data Set consists of the voting records of the 435 members of the U.S. House of Representatives in 1984 on 16 key votes identified by the *Congressional Quarterly Almanac* (CQA). This data set is a part of the UCI machine learning repository (Dheeru and Karra Taniskidou, 2017), and has been used previously for classification and filtering problems (He et al., 2002, 2003). The goal of this analysis is to determine which Representatives, if any, consistently deviated from their party of record, and to compare the ability of different algorithms to identify these individuals. This is different from the other classification filtering analyses in that there is no objective truth to discover: the party of record for each member cannot be judged "correct" or "incorrect". On a practical level, this means that it is impossible discuss the FOR or specificity of any algorithm, as these terms have no meaning in this context. For consistency in terminology, each member will still be "retained" or 'removed" by the algorithms, with the understanding that a "retained" Representative is one whose voting record is consistent with other members of his or her party, while a "removed" Representative is one whose voting record is inconsistent with the other members of their political party.

5.2 Methods

The data set consists of 17 columns, with one column recording the party of record ("Democrat" or "Republican"), and the remaining columns recording the vote outcome from the corresponding representative. These outcomes are simplified from the

Table 5.1: The 21 algorithms compared to classCleaner in the Congressional Voting data set. For the complete list of all comparative algorithms, see Appendix C.

Algorithms					
AENN	BBNR	CVCF	DCF	EB	EF
ENG	ENN	EWF	GE	HARF	HRF
I-C4.5	INFFC	IPF	MF	ORBoost	PF
PRISM	R-C4.5	V-C4.5			

original nine possible outcomes from the CQA to three: "yea" (voted for, paired for, and announced for), "nay" (voted against, paired against, and announced against), and "unknown" (voted present, voted present to avoid conflict of interest, and did not vote or otherwise make position known). To calculate the pairwise distance between the voting records of congressional members, the votes $\{v_{i1}, \ldots, v_{i16}\}$ of congressional member i were coded as

$$v_{ik} = \begin{cases} 1 & \text{congressional member } i \text{ voted 'yea' on issue } k \\ 0.5 & \text{the position of congressional member } i \text{ on issue } k \text{ is unknown} \\ 0 & \text{congressional member } i \text{ voted 'nay' on issue } k \end{cases}$$

for $i = 1, \ldots, 435$ and $k = 1, \ldots, 16$. The total distance between members i and j was then calculated using the Manhattan distance metric

$$d_{ij} = \text{dist}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{16} |\mathbf{x}_{ik} - \mathbf{x}_{jk}|,$$

so that the distance between two members increases by 1 when they vote opposite ways on an issue, and increases by 0.5 when one votes either 'yea' or 'nay' on the issue and the other position is unknown. The parameter α_0 was set to 0.05.

For the purposes of comparison, 21 different algorithms from the NoiseFiltersR package (Morales et al., 2016) in R were also applied, as listed in Table 5.1. All algorithms were run using the default parameters. In addition to removing instances

which are judged incorrect, some algorithms examine whether any instances can be "fixed" by reclassifying instances into another category if sufficient evidence exists. For the purposes of this analysis, these options are ignored, and all members who are reclassified are simply removed.

5.3 Results

To simplify the analysis, the voting records of each member were clustered using hierarchical clustering using Manhattan distance and multiple methods of agglomeration including complete, average, and the Ward method. Of the methods selected, the Ward method required the fewest clusters (three) to separate Democrats and Republicans, and was used to arrange the party membership, predicted membership, and voting record of each Representative in Figure 5.1. Of the three clusters, one is almost entirely composed of Democrats (209 Democrats out of 217 total members), one is almost entirely composed of Republicans (151 Republicans out of 160 total members). The last cluster is intermediate in nature – although it has a higher proportion of Democrats (49) than Republicans (9), this group appears to have a voting record more similar to that of Republicans than Democrats, as suggested by the order of the clustering and visually in the vote record.

Table 5.2 shows the proportion of representatives retained from each cluster using all filtering methods. Most algorithms performed extremely well in the Democratic cluster: eleven removed all Republican members, and only three algorithms retained more than three Republicans. All but four algorithms kept at least 191 of the 209 Democrats. ClassCleaner was one of eight algorithms to remove all eight Republicans while retaining 207 Democrats. In the Republican cluster, classCleaner was one of only four algorithms to keep a higher proportion of Republicans than Democrats and the only one to keep over 90 % of Republicans, keeping 138 Republicans and no Democrats. EF retained 94/151 Republicans and 5/9 Democrats, DCF retained

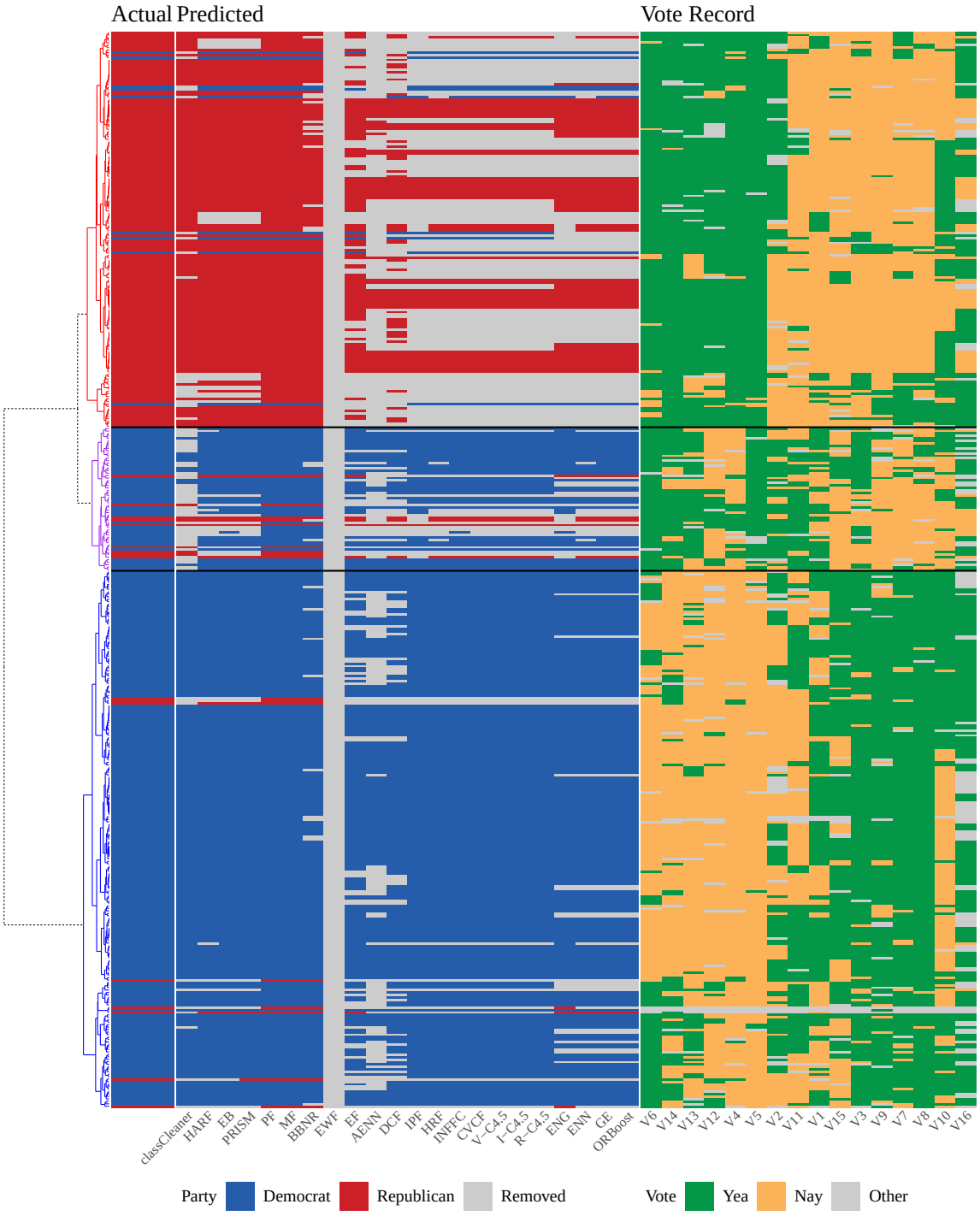


Figure 5.1: The voting data of each member of the Congress (rows) on 16 key votes (columns). The columns marked "Actual" shows the party of record for each Representative. The 22 columns marked "Predicted" are red/blue if the algorithm retains the observation as a Republican or Democrat respectively, and gray if the member is removed by the algorithms.

77/151 Republicans and 3/9 Democrats, while AENN removed all Democrats but only retained 44 Republicans. Twelve algorithms did not remove any Democrats, and of those not previously listed, only PRISM and EB retained over half of the Republicans. For these two clusters, the desired behavior of the filtering algorithm is straightforward, and only classCleaner selectively identified those members whose voting patterns did not match their party in across both groups.

The desired behavior of the filtering algorithms on Representatives assigned to the intermediate cluster is less clear. If this cluster is viewed as a subset of the Democrats (49 out of 58 cluster members are Democrats), then the desired outcome is to remove the Republicans while retaining the Democrats. Most of the explored algorithms had this effect: sixteen algorithms retained 65 % to 88 % of Democrats while removing at least 44 % of Republicans. Alternatively, these individuals could be viewed as outliers from both parties, in which case they should generally be removed from both parties. ClassCleaner was the only algorithm to provide this type of result – the seven retained Republicans and ten retained Democrats were those with voting records most similar to those of their respective parties. If the goal of the analysis is to identify the members whose voting patterns are consistent with the dominant groups in the party, only classCleaner achieves this objective.

Using the simulation data, classCleaner was shown to have better behavior when the instances were not evenly divided among groups, especially when the sample size is small. In this data set, the small sample size (16 votes) coupled with the disparity across groups (168 Republicans versus 267 Democrats) appears to make most algorithms far more likely to remove the smaller group of Republicans and retain Democrats. This shows the strength of using classCleaner in settings where the sample size is small.

Table 5.2: For each cluster (identified as Democratic, Intermediate, or Republican based on the proportion of members from each party included), the results of each algorithm on the Representatives from each party (D = Democrat, R = Republican). The numbers next to each cluster give the total number of members from the corresponding party who were initially included in the cluster.

	Democratic		Intermediate		Republican	
Algorithm	D (209)	R (8)	D (49)	R (9)	D (9)	R (151)
classCleaner	99 %	0 %	20 %	78 %	0 %	91 %
adaBoost						
EB	100 %	25 %	88 %	56 %	100 %	89 %
ORBoost	91 %	12 %	71 %	56 %	78 %	41 %
IBL						
AENN	63 %	0 %	53 %	11 %	0 %	29 %
BBNR	95 %	100 %	100 %	89 %	100 %	95 %
ENG	92 %	38 %	80 %	22 %	78 %	38 %
ENN	91 %	12 %	67 %	56 %	67 %	41 %
GE	91 %	12 %	71 %	56 %	78 %	41 %
Voting						
CVCF	99 %	0 %	80 %	44 %	100 %	30 %
DCF	85 %	0 %	65 %	44 %	33 %	51 %
EF	83 %	12 %	71 %	56 %	56 %	62 %
HARF	99 %	25 %	82 %	56 %	100 %	89 %
HRF	99 %	0 %	76 %	44 %	89 %	30 %
PF	100 %	100 %	100 %	100 %	100 %	100 %
V-C4.5	99 %	0 %	80 %	44 %	100 %	30 %
Iterative						
I-C4.5	99 %	0 %	80 %	44 %	100 %	30 %
INFFC	99 %	0 %	82 %	44 %	100 %	30 %
IPF	99 %	0 %	84 %	11 %	100 %	28 %
R-C4.5	99 %	0 %	80 %	44 %	100 %	30 %
Other						
EWF	0 %	0 %	0 %	0 %	0 %	0 %
MF	100 %	100 %	100 %	100 %	100 %	100 %
PRISM	100 %	38 %	86 %	56 %	100 %	89 %

Chapter 6

Discussion

LC-MS/MS has the potential to expand our understanding of the proteome through its ability to relatively quantify many proteins their proteoforms simultaneously. To maximize the potential of this technology, it is necessary to find and remove misidentified peptides that misrepresent the intensity of these proteins. This dissertation introduces classCleaner, a new algorithm designed to find mislabeled peptides in each protein, that is generic in the sense that it can be applied to other problems, but tailored to the particularities of the proteomics problem.

6.1 The characteristics of classCleaner

ClassCleaner is best described as a one-vs-all algorithm that validates the protein membership of each peptide, where validate is taken to mean that under the null hypothesis the peptides are assumed to belong to the protein to which they were originally assigned, and a significant test result is required to overturn that assessment. This approach requires very few assumptions, naturally yields a very high specificity, and can be implemented very easily and efficiently using statistical software.

6.1.1 Assumptions

ClassCleaner is completely non-parametric, and requires few assumptions regarding the original data. Assumption 1 merely assumes that peptides belonging to the same protein tend to be closer together than those from different proteins. Informally, this is a requirement of all classification and clustering algorithms, although defining "closer" is necessarily context-dependent.

The model also assumes that the set of distances originating from a given peptide are independent and identically distributed. Notably, distances are not independent when combining multiple rows or columns (even if shared distances are ignored), nor is this assumed. Our limited assumption of independence for each row (or column) is sufficient to obtain the empirical distributions of $\bar{G}$ and $\bar{F}$, which can still be combined to produce an overall estimate.

The last assumption made in classCleaner, that $G_i(t) \equiv G(t)$ (2.5), warrants additional discussion. A reasonable argument could be made that the distributions $G_1, \ldots, G_{N_1}$ should be treated as different, and estimated accordingly. This presents a problem when trying to estimate these distributions, specifically in the case where a peptide i is not identified correctly. For a misidentified peptides i, the estimate of G_i no longer represents within-peptide distances (because i is not a true member of the protein), but between-peptide distances, yielding a violation of Assumption 1. In fact, for $\alpha < 0.05$, replacing $\hat{\tau}$ with $\hat{\tau}_i = \hat{G}_i(t^*)$ yields a test that never removes any peptides. Assuming that most peptides are correctly identified, averaging all N_1 estimates of G_i provides a more stable estimate of τ.

6.1.2 Approach and Implementation

Because classCleaner takes correct identification as the null hypothesis, there needs to be sufficient evidence that peptides should be removed before the original assumption is overturned. Using a Bonferroni correction amplifies this effect, making it even more difficult to remove peptides. The effect of this was relative small, however, increasing the specificity by less than 0.07 for all spike-in proteins independent of benchmark, relative to using no correction at all. Interestingly, the Bonferroni correction had the least effect on the specificity of APOB, despite its larger size. While other corrections could be considered, any correction must be valid under arbitrary dependence conditions. In particular, the Benjamini-Hochburg method (Benjamini and Hochberg,

1995) is not valid under the dependence structure imposed on the set of Z_i from the distance matrix, and the Benjamini-Yekutieli method (Benjamini and Yekutieli, 2001) produced nearly identical results to the Bonferroni method.

One side-effect of analyzing each protein sequentially is the ease in which classCleaner can be adapted for peptides assigned to two or more groups. For example, consider an alternative analysis in which A1AG1 and A1AG2 are analyzed separately. The seven peptides shared between the two proteins can be analyzed with respect to both proteins. At present, this would be accomplished by rerunning the algorithm twice, with only a single protein, A1AG1 or A1AG2, analyzed during each run. The vector of protein identities would only need to distinguish between the 37 (27) peptides assigned to A1AG1 (A1AG2) versus all other protein in run. Future versions of the algorithm could provide additional options to accomplish this task from the perspective of the end-user, but the process would be the same. Notably, the conclusions regarding all other peptides not assigned to A1AG1 or A1AG2 are unaffected by this change.

ClassCleaner can be easily implemented using standard statistical software such as R (R Core Team, 2017), as shown in Appendix B. The run time for classCleaner is $O(N^2)$ given the pairwise distance matrix (where the run time is $O(N^2 n)$). The more computationally expensive calculations such as solving $phi(t^*) = t^*$ and calculating the inverse binomial are performed once per protein. This makes classCleaner a fast and efficient solution for validating class labels.

6.2 Alternative algorithms

ClassCleaner performs as well as or better than almost every algorithm to which it was compared. However, the algorithms from classification filtering, are not optimized for the same task. For these algorithms, the primary question is whether any of them can be adapted to the task required in proteomics. Conversely, the pro-

teomic filtering algorithms are designed to filter proteomic data. Furthemore, both alternative algorithms go a step further by clustering the peptides in each protein into proposed proteoforms. Thus, in addition to comparing the results of the filters, it is also necessary to address the relative utility of these proposed proteoforms.

6.2.1 Comparison to classification filters

Classification filtering algorithms present a possible source of ready-made solutions for removing misidentified peptides, and many are already implemented in R. However, these results suggest that because these algorithms are optimized towards improving the accuracy of classification analyses, they do not always improve upon the accuracy of the filtered data set. In particular, many algorithms are biased towards larger classes, especially when n is small. When filtering prior to a classification analysis, sacrificing specificity for a greater decrease in FOR is a logical choice, especially when the number of instances per class can be assumed to be approximately equal. Compared to these algorithms, classCleaner provides reliable and consistent behavior even when n is small ($n = 10$ in the simulations, and 16 votes in the congressional voting analysis) across classes of all sizes.

Of the classification algorithms tested, the EF algorithm provided comparable results to classCleaner when n was large. The EF algorithm is a voting algorithm in which the votes are obtained by applying a decision tree, neural network, and linear machine to the data sets created using four-fold cross validation. The principles behind this algorithm are general, and variations exist such as HRF and DCF. While these two particular algorithms included a neural network classification that required too much memory on the sickle-cell data set, they demonstrate the variety of algorithms that can be used to decide whether or not each peptide was labeled correctly.

6.2.2 Comparison to proteomic filtering algorithms

ClassCleaner, PQPQ, and BP-Quant are all designed to address challenges specific to proteomic data sets. In particular, algorithms used with LC-MS/MS need to work on data sets with hundreds of proteins, a wide range of peptide counts, heterogeneity within proteins, and correlation across proteins. To address the large number of proteins and the disparity across peptide counts, all three algorithms analyze a single protein at a time. The size of the protein can still affect the probability of keeping each peptide, as seen with both classCleaner and PQPQ on the simulation data, but these effects are far smaller than those observed for the classification filters. PQPQ and BP-Quant go a step further and only use the within-protein abundances to determine whether or not to keep each peptide, eliminating any concerns due to correlating proteins. In contrast, classCleaner incorporates distance information from other proteins in a manner that is insensitive to the relative size disparity between peptides in P_1 and those not in P_1. Furthermore, the correlation across proteins has a negligible effect so long as the number of peptides from highly correlated proteins is small relative to the overall number of proteins. When this condition is not met, it will generally be associated with small values of $\hat{\tau}$, which can be used as a sign that the initial assumptions of the algorithm are not met.

Both PQPQ and BP-Quant address heterogeneity within proteins by grouping peptides into proteoforms. In light of the fact that classCleaner does not group peptides, the proposed groups generated by PQPQ and BP-Quant were ignored in Chapter 4. While adding the capability to group peptides would be beneficial for classCleaner, and might be suggested as a reason to use PQPQ or BP-Quant over classCleaner, the proposed groups did not appear to add significant information to the analysis.

In both the simulation and the sickle-cell data, only the peptides included in the first cluster generated by PQPQ were retained because additional clusters contained

a very high proportion of incorrect peptides, This behavior is due to specific choices in the PQPQ algorithm and is likely to be exhibited in other data sets as well. While modifications to PQPQ could remove these issues, at present the best results from PQPQ are likely to be obtained by only retaining peptides in the first cluster. This negates any benefit of proposed clusters in PQPQ, and makes the practical use of PQPQ equivalent to classCleaner.

The groups proposed by BP-Quant are determined by the set of significant contrasts. For the sickle-cell data set, pairwise comparisons across the four groups were used to produce six total contrasts and the pattern of signficance across these contrasts were used to group the peptides into groups. It was common for peptides in noisier clusters (based on HC) to be non-significant in all contrasts or have an unusual set of significant contrasts. These peptides were then removed, making BP-Quant extremely effective at reducing the FOR of all five proteins. Among those peptides in cleaner clusters (again, based on HC), most peptides had a similar pattern of large and small p-values across all six contrasts. For the five proteins studied, while different peptides clearly have slightly differing abundance patterns over the subjects, these patterns did not correlate with the disease/treatment groups. However, the mean p-value for some contrasts was near 0.05 (the FDR cut-off used for this analysis). This split up proteins with highly similar patterns into multiple groups based on the choice of threshold. This is a common difficulty with using thresholds of this nature, and accounted for the differences between all groups retained by BP-Quant for each of the five proteins. Furthermore, it also accounts for the poor specificity of BP-Quant, as some of the removed peptides had patterns that were highly similar to the retained pattern except on one to two contrasts. For example, the retained peptides for CERU had significance patterns {1, 1, 1, 0, 0, 0} (50 peptides) or {0, 0, 1, 0, 0, 0} (67 peptides). The nine peptides with pattern {0, 1, 1, 0, 0, 0} and four peptides with pattern {1, 0, 1, 0, 0, 0} were removed. Additional peptides with

similar abundance patterns lacking any significant contrasts are also removed. The groups proposed by BP-Quant appear prone to these types of arbitrary division, and require manual review to correct. They also retain a set of peptides that are intentionally biased towards significance. While this is not necessarily problematic in a discovery-based study, it is important to remember when interpreting the results.

While all three algorithms were designed with the particularities of protoemic data sets in mind, only classCleaner is easily adaptable to other filtering problems. PQPQ assumes a confidence score exists on the reliability of each estimate, and also assumes that peptides with higher mean intensities are more likely to be identified correctly than those with lower mean intensities. BP-Quant assumes that the samples are split into groups, and that comparing the behavior across these groups is of primary interest. While these assumptions are all universally true in label-free shotgun proteomic studies, neither holds in general.

Ultimately, for all the similarities and differences between these algorithms, the most important difference between these algorithms is in their performance. Based on both the simulation and spike-in studies, classCleaner and PQPQ have similar behavior in terms of specificity and $\%\Delta$, with classCleaner having universally better results. Conversely, BP-Quant has a much greater decrease in the FOR at the cost of a much larger drop in specificity relative to both classCleaner and PQPQ. The choice of classCleaner or BP-Quant thus depends on the primary goals of the analysis, and whether the bias towards significant peptides in BP-Quant is detrimental.

6.3 Future Work

Further development of classCleaner is focused on two tasks: extending the algorithm to smaller values of N_1 and iterative methods. To extend the algorithm to smaller values of N_1, the estimate of $\bar{G}$ must be improved, most obviously by borrowing information across proteins. Extensions in this direction appear fairly straightforward,

but optimizing this task requires careful consideration of the order in which proteins are addressed and the relative weight given to each protein.

Iterative methods can be best described as sacrificing specificity in order to further decrease the FOR. They also have an advantage in that the estimates of τ and t^* are more accurate when the proportion of misidentified peptides is smaller. While several iterative methods developed for classification analyses were included in the comparison, the base procedures these methods iterated were universally biased towards larger classes. Iterating these methods magnified this bias, resulting in the removal of all or most peptides from smaller proteins. Although classCleaner lacks this bias, there is still a risk of removing too many peptides, especially for larger proteins where heterogeneity is more common. One possible iterative method, where the procedure removes only the most significant peptide before re-estimating τ and t^* removes too many peptides due to the correlation between distances. In practice, iterative solutions need to account for the dependence among the test statistics in a way that the current procedure does not do.

Appendix A

Technical Details and Proofs

This appendix provides the technical details and proofs of the lemmas in Chapter 2.

Lemma 1 shows that for continuous CDFs $\bar{G}$ and $\bar{F}$ defined in (2.5) and (2.7) respectively, there is a unique point where $\bar{G}(t) = 1 - \bar{F}(t)$, or equivalently, where $t = \bar{G}^{-1}\left(1 - \bar{F}(t)\right)$ assuming this later function is defined. By definition, every CDF is monotonically non-decreasing with $F(-\infty) = 0$ and $F(\infty) = 1$. Consequently, the functions $\bar{G}(t)$ and $1 - F(t)$ must cross at least once. Noting that $\psi(t)$ is only defined where $\bar{G}$ monotonically increasing, if either $\bar{G}$ or $\bar{F}$ has jump points, it is possible that the crossing occurs at a jump, such that there is no point at which $\bar{G}(t) = 1 - \bar{F}(t)$. Thus, both functions must be continuous. Formally, this is proved as follows.

Lemma 1. *Define $\psi(t) = \bar{G}^{-1}(1 - \bar{F}(t))$ as in (2.13), and set $h(t) = \psi(t) - t$. Then $h(t)$ has a unique root at t^* at which*

$$\psi(t^*) = t^*.$$

Proof. First, note trivially that for $t < 0$, $\bar{G}(t) = \bar{F}(t) \equiv 0$. Thus,

$$\begin{aligned} \lim_{t\to 0} h(t) &= \lim_{t\to 0} \bar{G}^{-1}(1 - \bar{F}(t)) - t & \lim_{t\to \infty} h(t) &= \lim_{t\to \infty} \bar{G}^{-1}(1 - \bar{F}(t)) - t \\ &= \bar{G}^{-1}(1 - 0) - 0 & &= \bar{G}^{-1}(1 - 1) - \infty \\ &= \infty & &= -\infty \end{aligned}$$

and there is at least one point t^* such that $h(t^*) = 0$. To show that this point is unique, consider

$$\begin{aligned} h(t) &= \psi(t) - t \\ h(t) + t &= \bar{G}^{-1}(1 - \bar{F}(t)) \\ \bar{G}(h(t) + t) &= 1 - \bar{F}(t) \end{aligned} \tag{A.1}$$

Taking the derivative of (A.1) with respect to t gives

$$\begin{aligned} \frac{d}{dt}\bar{G}(h(t) + t) &= \frac{d}{dt}(1 - \bar{F}(t)) \\ g(h(t) + t)(h'(t) + 1) &= -f(t) \\ h'(t) &= -\frac{f(t)}{g(h(t) + t)} - 1 < 0. \end{aligned}$$

for all $t \in \mathbb{R}$ such that $g(h(t)+t) > 0$. Since the slope of $h(t)$ is always negative, $h(t)$ is a monotonically decreasing function, and the solution t^* must be unique. □

Lemmas 2 and 3 show that Z_i is the sum of independent Bernoulli random variables, with probability τ or $1-\tau$ depending on whether or not $Z_i \in P_1$ (by assumption, for $j = 1, \ldots, i-1, i+1, \ldots, N_1$, $j \in P_1$). Independence of $d_{i1}, \ldots, d_{i(i-1)}, d_{i(i+1)}, \ldots, d_{iN_1}$ is a consequence of (2.4).

Lemma 2. *If $i \in P_1$, then Z_i is a sum of i.i.d, r.v.s, and for each $i = 1, 2, \ldots, N_1$,*

$$Z_i|_{\mathcal{H}_0^{(i)}} \sim Bin\left(N_1 - 1, \bar{G}(t^*)\right) \equiv Bin(N_1 - 1, \tau).$$

such that $\mathrm{E}\left[Z_i \middle| \mathcal{H}_0^{(i)}\right] = (N_1 - 1)\bar{G}(t^*)$.

Proof. For all $i, j = 1, \ldots, N_1$ and $i \neq j$,

$$\Pr\left(\mathrm{I}[d_{ij} \leq t^*] = 1\right) = \Pr(d_{ij} \leq t^*) = \bar{G}(t^*) = \tau$$

from (2.14). Now, fix i, such that $i \in \{1, \ldots, N_1\}$. By (2.4), $\mathrm{I}[d_{ij} \leq t^*]$ are independent over $j = 1, \ldots, i-1, i+1, \ldots, N_1$. Thus, $Z_i = \sum_{j=1,\ j\neq i}^{N_1} \mathrm{I}[d_{ij} \leq t^*]$ is the sum of

independent Bernoulli random variables with probability of success τ, and thus has a binomial distribution. Consequently,

$$\mathrm{E}[Z_i] = (N_1 - 1)\tau = (N_1 - 1)\bar{G}(t^*).$$

□

Lemma 3. *If $i \notin P_1$, but $j \in P_1$ for $j = 1, \ldots, i-1, i+1, \ldots, N_1$, then Z_i is a sum of independent r.v.s with distribution*

$$Z_i|_{\mathcal{H}_1^{(i)}} \sim Bin(\big(N_1 - 1, \bar{F}(t^*)\big) \equiv Bin(N_1 - 1, 1 - \tau)$$

where $\mathrm{E}\left[Z_i \,\middle|\, \mathcal{H}_1^{(i)}\right] = (N_1 - 1)\bar{F}(t^*)$.

Proof. For any $j = 1, \ldots, i-1, i+1, \ldots, N_1$ and $i \notin P_1$,

$$\Pr\left(\mathrm{I}[d_{ij} \leq t^*] = 1\right) = \Pr(d_{ij} \leq t^*) = \bar{F}(t^*) = 1 - \tau$$

from (2.9) and (2.14). For a fixed i, $\mathrm{I}[d_{ij} \leq t^*]$ are independent over $j = 1, \ldots, i-1, i+1, \ldots, N_1$ by (2.6). Thus, $Z_i = \sum_{j=1,\ j\neq i}^{N_1} \mathrm{I}[d_{ij} \leq t^*]$ is the sum of independent Bernoulli random variables with probability of success $1 - \tau$, and thus has a binomial distribution. Consequently,

$$\mathrm{E}[Z_i] = (N_1 - 1)(1 - \tau) = (N_1 - 1)\bar{F}(t^*).$$

□

By setting $\tau^* = G(t) = 1 - F(t)$, there is a symmetry in the distribution of Z_i under the null and alternative hypotheses. Lemmas 4 - 6 highlight the effects of that symmetry. In particular, Lemma 5 shows that when $a_\alpha \geq \frac{N_1 - 1}{2}$, then the type I error rate is larger than the type II error rate. At $\frac{N_1 - 1}{2}$, the distribution of Z_i under the null and alternative hypothesis is equal, because the two distributions are symmetric

about that point. For any $z > \frac{N_1-1}{2}$, $b(z, N_1 - 1, \tau) > b(z, N_1 - 1, 1 - \tau)$. Lemma 6 shows that control is obtained in both directions under these circumstances.

Lemma 4. *Suppose $X \sim Bin(n, p)$ and $Y \sim Bin(n, 1-p)$. For any α, let a_α and b_α be selected according to*

$$a_\alpha := \arg\max_x \{\Pr(X \leq x | \, p) \leq \alpha\} \tag{2.21}$$

and

$$b_\alpha := \arg\min_y \{\Pr(Y \geq y | \, 1-p) \leq \alpha\} \tag{2.22}$$

Then $b_\alpha = n - a_\alpha$.

Proof. At b_α,

$$\alpha \geq \Pr(Y \geq b_\alpha | \, 1-p) = \Pr(X \leq n - b_\alpha | \, p)$$

by the properties of the binomial distribution. Let $a^* = n - b_\alpha$. Clearly, $a^* \leq a_\alpha$, because a_α is the largest value at which (2.21) holds. Similarly, let $b^* = n - a_\alpha$. Then $b^* \leq n - a^* = b_\alpha$. But

$$\alpha \geq \Pr(X \leq a_\alpha | \, p) = \Pr(X \leq n - b^* | \, p) = \Pr(Y \geq b^* | \, 1-p),$$

so by (2.22), $b^* \geq \beta_\alpha$. Consequently, $b_\alpha = b^* = n - a_\alpha$, and $a_\alpha + b_\alpha = n$. □

Lemma 5. *Suppose for a fixed α, the test is conducted using a cut-off $a_\alpha \geq \frac{N_1-1}{2}$ that satisfies (2.16). Let $\beta(\alpha)$ be the corresponding type II error of the test such that*

$$\beta(\alpha) = \Pr(Z_i \geq a_\alpha | \mathcal{H}_1^{(i)}).$$

Then $\beta(\alpha) \leq \alpha$.

Proof. Since $a_\alpha \geq \frac{N_1-1}{2}$, then $N_1 - 1 - a_\alpha \leq \frac{N_1-1}{2}$. Consequently,

$$\beta(\alpha) = \Pr(Z_i \geq a_\alpha | \mathcal{H}_1^{(i)}) = \Pr(Z_i \geq a_\alpha | 1 - \tau)$$

$$= \Pr(Z_i \leq N_1 - 1 - a_\alpha | \tau)$$
$$\leq \Pr(Z_i \leq a_\alpha | \tau)$$
$$= \alpha.$$

□

Lemma 6. *Suppose for a fixed β, the test is conducted using a cut-off $b_\beta \leq \frac{N_1 - 1}{2}$ that satisfies (2.20). Let $\alpha(\beta)$ be the corresponding type I error of the test,*

$$\alpha(\beta) = \Pr(Z_i \leq b_\beta | \mathcal{H}_0^{(i)}).$$

Then $\alpha(\beta) \leq \beta$.

Proof. This follows using the same proof in Lemma 5, with appropriate changes in the direction of the inequalities. □

Lemma 7. *Let a_α be defined according to (2.16) for some fixed $\alpha < 0.5$ and $\tau > \frac{1}{2}$ (Assumption 1). Let $z_\alpha = \Phi^{-1}(\alpha)$ be the α^{th} percentile of the standard normal distribution, and define*

$$\tau^* = \frac{1}{2} + \frac{1}{2}\sqrt{\frac{z_\alpha^2}{z_\alpha^2 + N_1 - 1}}. \tag{2.23}$$

Then assuming the conditions for the binomial approximation hold, (e.g. $\min(N_1\tau, N_1\tau(1-\tau)) > 5$, (Schader and Schmid, 1989)), and $\tau \geq \tau^*$, it follows that $\alpha \geq \beta$ where

$$\beta = \Pr(Z_i > a_\alpha | 1 - \tau).$$

Proof. For simplicity, set $n = N_1 - 1$. Define

$$X \sim N(\mu = n\tau, \sigma^2 = n\tau(1-\tau))$$
$$Y \sim N(\mu = n(1-\tau), \sigma^2 = n\tau(1-\tau))$$

By the normal approximation to the binomial,

$$\Pr(X \leq x) \approx \Pr\left(Z_i \leq x \,|\, Z_i \sim Bin(n, \tau)\right)$$

and

$$\Pr(Y \leq x) \approx \Pr\left(Z_i \leq x \,|Z_i \sim Bin(n, 1-\tau)\right)$$

such that

$$\alpha \geq \Pr(X \leq a_\alpha^*) = \Pr\left(\frac{X - n\tau}{\sqrt{n\tau(1-\tau)}} \leq \frac{a_\alpha - n\tau}{\sqrt{n\tau(1-\tau)}}\right) = \Phi\left(\frac{a_\alpha - n\tau}{\sqrt{n\tau(1-\tau)}}\right).$$

and

$$\beta = \Pr(Y > a_\alpha^*) = 1 - \Phi\left(\frac{a_\alpha - n(1-\tau)}{\sqrt{n\tau(1-\tau)}}\right)$$

Now, define

$$z_\alpha = \frac{a_\alpha - n\tau}{\sqrt{n\tau(1-\tau)}} \qquad\qquad z_{1-\beta} = \frac{a_\alpha - n(1-\tau)}{\sqrt{n\tau(1-\tau)}}$$

and note that

$$z_{1-\beta} = \frac{a_\alpha - n(1-\tau)}{\sqrt{n\tau(1-\tau)}} = \frac{a_\alpha + n\tau - n}{\sqrt{n\tau(1-\tau)}} = \frac{a_\alpha - n\tau + 2n\tau - n}{\sqrt{n\tau(1-\tau)}} = z_\alpha - \frac{n + 2n\tau}{\sqrt{n\tau(1-\tau)}}$$

Consequently,

$$\begin{aligned}
\frac{n + 2n\tau}{\sqrt{n\tau(1-\tau)}} &= z_\alpha - z_{1-\beta} \\
n(1+2\tau) &= (z_\alpha - z_{1-\beta})\sqrt{n\tau(1-\tau)} \\
n(1+2\tau)^2 &= (z_\alpha - z_{1-\beta})^2(\tau - \tau^2) \\
0 &= \left[(z_\alpha - z_{1-\beta})^2 + 4n\right]\tau^2 - \left[(z_\alpha - z_{1-\beta})^2 + 4n\right]\tau + n
\end{aligned}$$

By the symmetry of the normal distribution, $z_{1-\beta} = -z_\beta$, so the roots of this quadratic equation are

$$\begin{aligned}
\tau &= \frac{(z_\alpha + z_\beta)^2 - 4n \pm \sqrt{\left[(z_\alpha + z_\beta)^2 + 4n\right]^2 - 4\left[(z_\alpha + z_\beta)^2 + 4n\right]n}}{2\left[(z_\alpha + z_\beta)^2 + 4n\right]} \\
&= \frac{1 \pm \sqrt{1 - 4\frac{n}{(z_\alpha + z_\beta)^2 + 4n}}}{2} \\
&= \frac{1}{2} \pm \frac{1}{2}\sqrt{\frac{(z_\alpha + z_\beta)^2 + 4n - 4n}{(z_\alpha + z_\beta)^2 + 4n}} = \frac{1}{2} \pm \frac{1}{2}\sqrt{\frac{(z_\alpha + z_\beta)^2}{(z_\alpha + z_\beta)^2 + 4n}}
\end{aligned}$$

From (2.23), $\tau \geq \tau^*$, so

$$\frac{(z_\alpha + z_\beta)^2}{(z_\alpha + z_\beta)^2 + 4n} \geq \frac{z_\alpha^2}{z_\alpha^2 + n} = \frac{(2z_\alpha)^2}{(2z_\alpha)^2 + 4n}. \tag{A.2}$$

Since $\alpha < 0.5$, $z_\alpha < 0$, so (A.2) is only true for $z_\beta \leq z_\alpha$, and thus $\beta < \alpha$. □

Lemma 8. *Under the global null hypothesis, $\hat{\tau}$ is an unbiased estimator of τ. That is,*

$$\mathrm{E}\left[\hat{\tau}|\mathcal{H}_0^*\right] = \tau$$

Proof. Under the global null hypothesis, all N_1 peptides are identified correctly. Consequently,

$$Z_i|\mathcal{H}_0^* \sim Bin(N_1 - 1, \tau)$$

Thus, it hold that

$$\begin{aligned}
\mathrm{E}[\hat{\tau}|\mathcal{H}_0^*] &= \mathrm{E}\left[\frac{1}{N_1}\sum_{i=1}^{N_1}\frac{Z_i}{N_1 - 1}\middle|\mathcal{H}_0^*\right] \\
&= \frac{1}{N_1(N_1-1)}\sum_{i=1}^{N_1}\mathrm{E}\left[Z_i|\mathcal{H}_0^*\right] \\
&= \frac{1}{N_1(N_1-1)}\sum_{i=1}^{N_1}(N_1 - 1)\tau \\
&= \tau
\end{aligned}$$

□

Appendix B

classCleaner Code

The classCleaner package exports three functions: classCleaner and two additional functions used to simulate data.

B.1 classCleaner

ClassCleaner was implemented using R (R Core Team, 2017). The required input is a matrix of distances, D, and the initial assignment, assignment. In addition, the classes parameter allows the algorithm to be run on only a subset of proteins. By default, this is set to "all", and the algorithm is applied to all proteins with sufficient peptides (controlled by min_count). Although the algorithm reports a p-value for each peptide, the value of a is provided in the output for a given value of alpha0. When the distance matrix does not have row and column labels, the label option allows labels to be assigned. It requires no external dependencies in R, and only one function defined within the package, psi, which is described next.

```
1 classCleaner <- function(D, assignment, classes = 'all',
    alpha0 = 0.05, labels = NULL, min_count = 20) {
2
3   # Check to make sure distance matrix is a symmetric, non-
      negative definite matrix and we have an assignment for
      each entry.
4   if(!(is.matrix(D) && isSymmetric(D) && is.numeric(D))) stop
      ("D must be a symmetric matrix with numeric entries")
```

```
5   if(length(assignment) != nrow(D)) stop("length(asssignment)
        != nrow(D)")
6   if(min(D) < 0) stop("D should be a distance matrix with
       entries >= 0.")
7   if(length(alpha0) != 1) {
8     warning("Multiple values of alpha0 found.  Only the first
          is used.")
9     alpha0 <- alpha0[1]
10  }
11
12  class_table <- table(assignment)
13
14  if(!identical(classes, "all")) {
15    Nk <- tryCatch({
16      Nk.tmp <- class_table[names(class_table) %in% classes]
17    },
18      error = function(cond){
19        message("An error ocurred in selecting which classes
              to filter.")
20        message("Here's the original error message:")
21        message(cond)
22      },
23      warning = function(cond){
24        message("A problem ocurred in selecting which classes
              to filter.")
25        message("Here's the original warning message:")
26        message(cond)
```

```
27        })
28    }
29    else {Nk <- class_table}
30
31    # handle labels
32    if(is.null(labels)){
33      if(is.null(rownames(D))){
34        if(is.null(colnames(D))) labels <- 1:ncol(D)
35        else labels <- colnames(D)
36      } else labels <- rownames(D)
37    }
38
39    result <- lapply(names(Nk)[min_count], function(k){
40
41      D11 <- D[which(assignment == k), which(assignment == k)]
42      D21 <- D[which(assignment == k), which(assignment != k)]
43
44      alpha <- alpha0 / Nk[k]
45
46
47      psi_t <- psi(D11[lower.tri(D11)], D21)
48
49      Zi_psi <- data.frame(
50        Zi = vapply(1:Nk[k], function(i) sum(D11[-i,i] < psi_t[
              "t"]), 0),
51        instance = labels[assignment == k],
52        index  = which(assignment == k)
```

```
53      )
54      Zi_psi <- within(Zi_psi[order(Zi_psi$Zi),], {
55        a <- stats::qbinom(alpha, Nk[k] - 1, psi_t["tau"]) - 1
56        tau_hat <- Zi / Nk[k]
57        p <- stats::pbinom(Zi, Nk[k] - 1, psi_t["tau"])
58
59        t <- psi_t["t"]
60        tau <- psi_t["tau"]
61        alpha0 <- alpha0
62        Nk <-  as.numeric(Nk[k])
63        k <- factor(k)
64      })
65    })
66    result <- do.call("rbind", result)
67 }
```

For increased speed and efficiency, the psi function was coded using Rcpp (Eddelbuettel and François, 2011; Eddelbuettel, 2013; Eddelbuettel and Balamuta, 2017). This function returns $(t^*, \hat{\tau})$ such that $\hat{\tau} = \hat{\bar{G}}(t^*) = 1 - \hat{\bar{F}}(t^*)$, as described in (2.27) and (2.28).

```
1 #include <RcppArmadillo.h>
2 // [[Rcpp::depends(RcppArmadillo)]]
3 using namespace Rcpp;
4
5 // [[Rcpp::export]]
6 NumericVector psi(const NumericVector& x, const NumericVector
      & y) {
7   int Nx = x.size();
```

```
8   int Ny = y.size();
9
10  double inc_x = 1. / Nx;
11  double inc_y = 1. / Ny;
12  double fin = inc_x * inc_y;
13
14  arma::vec z = join_cols(as<arma::vec>(x), as<arma::vec>(y))
        ;
15  arma::vec cat = join_cols(arma::zeros(Nx), arma::ones(Ny));
16  arma::uvec indices = sort_index(z);
17  arma::uvec::iterator ind_ptr = indices.end();
18
19  double pF = 0;
20  double pG = 1;
21
22  do {
23    ind_ptr--;
24    if(cat(*ind_ptr) < .5) {
25      pG = pG - inc_x;
26    } else {
27      pF = pF + inc_y;
28    }
29  }
30  while(pG - pF > fin);
31
32  NumericVector result = NumericVector::create(_["t"] = (z(*
      ind_ptr) + z(*(ind_ptr + 1))) / 2, _["tau"] = pG);
```

```
33
34    return result;
35 }
```

B.2 Simulation

To facilitate data generation, the following functions are used to generate data using the two simulation methods described in Chapter 3.

Both methods are called by the simulate_clustered_data function in R, which calls one of two algorithms written in Rcpp to produce the results.

```
1 simulate_clustered_data <- function(
2    n = 100,          # total number of observations per instance
3    Nk = c(40,200), # number of instances in each group
4    s = c(1,1),
5    rho = matrix(c(.6,.1,.1,.25),nrow = 2,ncol = 2),
6    tau = 1,
7    method = c("by-class", "by-instance")
8 ) {
9    method = match.arg(method)
10
11   # error checking
12   if (length(s) == 1) s <- rep(s, length(Nk))
13   if(!is.matrix(rho)) rho <- as.matrix(rho)
14
15   if (nrow(rho) != ncol(rho)) stop("rho must be a square
        matrix.")
16
```

```
17
18  if (length(Nk) != length(s)) stop("s must be of length 1 or
        have the same length as Nk")
19  if (length(tau) > 1) stop ("support for tau > 1 is not yet
       implemented.")
20  if (!all.equal(rho[lower.tri(rho)], rho[upper.tri(rho)]))
21    warning("rho is assumed to be a symmetric matrix.  Only
         the lower triangle is used.")
22
23  # compute it
24  if(method == "by-instance"){
25    if (length(Nk) != nrow(rho)) stop("Nk must have the same
         length as ncol(rho)/nrow(rho)")
26    X <- sim_by_instance(n, Nk, s, rho)
27  }
28  else{
29    if (length(Nk) == nrow(rho)){
30      X <- sim_by_class(n, Nk[Nk > 0], s[Nk > 0], tau, rho[Nk
           > 0, Nk > 0])
31    } else if (nrow(rho) == 1) {
32      X <- sim_by_class(n, Nk[Nk > 0], s[Nk > 0], tau, as.
         matrix(rho))
33    } else       stop("Nk must have the same length as ncol(
       rho)/nrow(rho) or rho must be of length 1.")
34
35
36  }
```

```
37
38   # add identifiers
39   rownames(X) <- paste("n",1:n,sep = ".")
40   colnames(X) <- paste0("N",rep(1:length(Nk),Nk),".",
41     unlist(sapply(Nk, function(x) {if(x > 0) 1:x else integer
         (0)})))
42
43   X
44 }
```

Simulating data using the "by-peptide" method.

```
1 #include <RcppArmadillo.h>
2 // [[Rcpp::depends(RcppArmadillo)]]
3 #include "mvrnorm.h"
4
5 // [[Rcpp::export]]
6 arma::mat sim_by_instance(arma::uword n, const arma::uvec& Nk
    , const arma::vec& s, const arma::mat& rho){
7
8   arma::uword K = Nk.n_elem;
9   arma::uword N = arma::sum(Nk);
10  double val;
11
12  arma::uvec K_range = arma::cumsum(Nk);
13  arma::mat V(N, N);
14
15  // generate variance-covariance matrix:
16  arma::uword k_start = 0, j_start = 0;
```

```
17  for(arma::uword k = 0; k < K; k ++) {
18    if(k == 0) k_start = 0;
19    else k_start = K_range(k - 1);
20
21    for(arma::uword j = 0; j <= k; j++) {
22      if(j == k){
23        val = rho(k,k) * s(k) * s(k);
24        V.submat(k_start, k_start, arma::size(Nk(k), Nk(k))).
              fill(val);
25      }
26      else {
27        if(j == 0) j_start = 0;
28        else j_start = K_range(j - 1);
29
30        val = s(k) * s(j) * rho(k, j);
31
32        V.submat(k_start, j_start, arma::size(Nk(k), Nk(j))).
              fill(val);
33        V.submat(j_start, k_start, arma::size(Nk(j), Nk(k))).
              fill(val);
34      }
35
36    }
37  }
38  V.diag().ones();
39
40  arma::vec mu(N, arma::fill::zeros);
```

```
41
42    arma::mat X = mvrnorm(n, mu, V);
43
44 return X;
45
46 }
```

Simulating data using the "by-protein" method.

```
 1 #include <RcppArmadillo.h>
 2 // [[Rcpp::depends(RcppArmadillo)]]
 3 #include "mvrnorm.h"
 4
 5 using namespace Rcpp;
 6
 7 //' @title tmp function
 8 //' @export
 9 // [[Rcpp::export]]
10 arma::mat sim_by_class( arma::uword n, const arma::uvec& Nk,
       const arma::colvec& s, double tau, const arma::mat& rho){
11
12    arma::uword K = Nk.n_elem;
13    arma::uword N = sum(Nk);
14    arma::mat V(K, K); // = s * s.t();
15    // S.diag() = s;
16
17    if(rho.n_elem == 1){
18      V.fill(arma::as_scalar(rho));
19      V.diag().ones();
```

```
20
21    }
22    else {
23      V = rho;
24    }
25
26    V = V % (s * s.t());
27
28    arma::vec mu0(K, arma::fill::zeros);
29    arma::mat W = mvrnorm(n,mu0, V);
30
31
32    arma::mat X = arma::randn<arma::mat>(n, N) * tau;
33
34    arma::uword j = 0;
35    arma::uword k = 0;
36    for(arma::uword i = 0; i < N; ++i, ++j){
37      if(j == Nk[k]) {
38        ++k;
39        j = 0;
40      }
41      X.col(i) += W.col(k);
42
43    }
44
45    return X;
46
```

```
47 }
```

Draws from a multivariate normal distribution were computed using the Cholesky decomposition.

```
1 #include <RcppArmadillo.h>
2 // [[Rcpp::depends(RcppArmadillo)]]
3
4 arma::mat mvrnorm(int n, arma::vec mu, arma::mat sigma) {
5   int ncols = sigma.n_cols;
6   arma::mat Y = arma::randn(n, ncols);
7   return arma::repmat(mu, 1, n).t() + Y * arma::chol(sigma);
8 }
```

Appendix C

Complete list of comparative algorithms

Table C.1: All comparative algorithms used in these analyses.

Algorithm	Shorthand	Citation
Proteomics Filters		
Protein quantification by peptide quality control	PQPQ	Forshed et al. (2011)
Bayesian proteoform quantification	BP-Quant	Webb-Robertson et al. (2014)
adaBoost filters		
Outlier removal boosting	ORBoost	Karmaker and Kwek (2005)
Edge boosting filter	EB	Wheway (2001)
Instance based learning filters		
All-k edited nearest neighbors	AENN	Tomek (1976)
Blame based noise reduction	BBNR	Delany and Cunningham (2004)
Editing with neighboring graphs	ENG	Sánchez et al. (1997)
Edited nearest neighbor	ENN	Wilson (1972)
Generalized edition	GE	Koplowitz and Brown (1981)
Voting Filters		
Cross-validated committees filter	CVCF	Verbaeten and Assche (2003)
Dynamic classification filter	DCF	Garcia et al. (2012)
Ensemble filter	EF	Brodley and Friedl (1996b,a, 1999)
High agreement random forest	HARF	Sluban et al. (2010)
Hybrid repair-remove filter	HRF	Miranda et al. (2009)
Partition filter	PF	Zhu et al. (2003)
Voting C4.5 filter	V-C4.5	Verbaeten (2002)

Table C.1: (continued)

Algorithm	Shorthand	Citation
Iterative filters		
Iterative C4.5 filter	I-C4.5	Verbaeten (2002)
Iterative noise filter based on the fusion of classifiers	INFFC	Sáez et al. (2014)
Iterative partitioning filter	IPF	Khoshgoftaar and Rebours (2007)
Robust C4.5 filter	R-C4.5	Verbaeten (2002)
Other filters		
Edge weight filter	EWF	Muhlenbach et al. (2004)
Mode filter	MF	Du and Urahama (2009, 2011)
Preprocessing instances that should be misclassfied	PRISM	Smith and Martinez (2011)

BIBLIOGRAPHY

Aebersold, R. and M. Mann (2003). Mass spectrometry-based proteomics. *Nature 422*(6928), 198–207.

Aha, D. W. and D. Kibler (1989). Noise-Tolerant Instance-Based Learning Algorithms. *Proceedings of the 11th international joint conference on Artificial intelligence (IJCAI'89) - Volume 1 1*, 794–799.

Benjamini, Y. and Y. Hochberg (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological) 57*(1), 289–300.

Benjamini, Y. and D. Yekutieli (2001). The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics 29*(4), 1165–1188.

Brodley, C. E. and M. Friedl (1996a). Improving Automated Land Cover Mapping by Identifying and Eliminating Mislabeled Observations from Training Data. In *Geoscience and Remote Sensing Symposium, 1996. IGARSS '96. 'Remote Sensing for a Sustainable Future.', International*, pp. 1379–1381 vol.2.

Brodley, C. E. and M. a. Friedl (1996b). Identifying and eliminating mislabeled training instances. In *Proceedings of the National Conference on Artificial Intelligence*, Portland, Oregon, pp. 799–805. AAAI Press ©1996.

Brodley, C. E. and M. A. Friedl (1999). Identifying Mislabeled Training Data. *Journal of Artificial Intelligence Research 11*(i), 131–167.

Clough, T., M. Key, I. Ott, S. Ragg, G. Schadow, and O. Vitek (2009, nov). Protein quantification in label-free LC-MS experiments. *Journal of Proteome Research 8*(11), 5275–5284.

Cover, T. M. and P. E. Hart (1967). Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory 13*(1), 21–27.

Craig, R. and R. C. Beavis (2004, jun). TANDEM: matching proteins with tandem mass spectra. *Bioinformatics 20*(9), 1466–7.

Delany, S. J. and P. P. P. Cunningham (2004). An Analysis of Case-Base Editing in a Spam Filtering System. In P. Funk and G. Calero (Eds.), *Advances in Case-Based Reasoning (Proceedings of the 7th. European Conferenc e on Case Based Reasoning, ECCBR-04)*, pp. 128–141. Springer Berlin Heidelberg.

Dheeru, D. and E. Karra Taniskidou (2017). {UCI} Machine Learning Repository.

Du, W. and K. Urahama (2009). Error-Correcting Semi-Supervised Learning with Mode-Filter on Graphs. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 2095–2100.

Du, W. and K. Urahama (2011). Error-Correcting semi-supervised pattern recognition with mode filter on graphs. *Journal of Advanced Computational Intelligence and Intelligent Informatics 15*(9), 1262–1268.

Eddelbuettel, D. (2013). *Seamless {R} and {C++} Integration with {Rcpp}*. New York: Springer.

Eddelbuettel, D. and J. J. Balamuta (2017, aug). Extending extit{R} with extit{C++}: A Brief Introduction to extit{Rcpp}. *PeerJ Preprints 5*, e3188v1.

Eddelbuettel, D. and R. François (2011). {Rcpp}: Seamless {R} and {C++} Integration. *Journal of Statistical Software 40*(8), 1–18.

Elias, J. E. and S. P. Gygi (2007). Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nature Methods 4*, 207–214.

Elias, J. E. and S. P. Gygi (2010, jan). Target-decoy search strategy for mass spectrometry-based proteomics. *Methods in molecular biology (Clifton, N.J.) 604*, 55–71.

Esary, J. D., F. Proschan, and D. W. Walkup (1967). Association of Random Variables, with Applications. *The Annals of Mathematical Statistics 38*(5), 1466–1474.

Forshed, J., H. J. Johansson, M. Pernemalm, R. M. M. Branca, A. Sandberg, and J. Lehtiö (2011). Enhanced Information Output From Shotgun Proteomics Data by Protein Quantification and Peptide Quality Control (PQPQ). *Molecular & Cellular Proteomics 10*(10), M111.010264.

Frénay, B. and M. Verleysen (2014). Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems 25*(5), 845–869.

Freund, Y. and R. E. Schapire (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences 55*(1), 119–139.

Garcia, L. P. F., A. C. Lorena, and A. C. Carvalho (2012). A study on class noise detection and elimination. In *2012 Brazilian Symposium on Neural Networks*, Curitiba, pp. 13–18. IEEE.

Grubbs, F. E. (1969). Procedures for Detecting Outlying Observations in Samples. *Technometrics 11*(1), 1–21.

He, Z., S. Deng, and X. Xu (2002). Outlier detection integrating semantic knowledge. In *Proceedings of the WAIM02*, pp. 126–131.

He, Z., X. Xu, and S. Deng (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters 24*(9-10), 1641–1650.

Henao, R., J. W. Thompson, M. A. Moseley, G. S. Ginsburg, L. Carin, and J. E. Lucas (2013). Latent protein trees. *Annals of Applied Statistics 7*(2), 691–713.

Hubert, M., P. J. Rousseeuw, and S. Van Aelst (2008). High-Breakdown Robust Multivariate Methods. *Statistical Science 23*(1), 92–119.

John, G. H. (1995). Robust decision trees: Removing outliers from databases. In *First International Conference on Knowledge Discovery and Data Mining*, Montreal, Qu\'ebec, Canada, pp. 174–179. AAAI Press.

Karmaker, A. and S. Kwek (2005). A boosting approach to remove class label noise. *Proceedings - HIS 2005: Fifth International Conference on Hybrid Intelligent Systems 2005*, 206–211.

Keller, A., J. Eng, N. Zhang, X.-j. Li, and R. Aebersold (2005). A uniform proteomics MS/MS analysis platform utilizing open XML file formats. *Molecular systems biology 1*, 2005.0017.

Khoshgoftaar, T. M. and P. Rebours (2004). Generating multiple noise elimination filters with the ensemble-partitioning filter. In *Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on*, Las Vegas, pp. 369–375. IEEE.

Khoshgoftaar, T. M. and P. Rebours (2007). Improving software quality prediction by noise filtering techniques. *Journal of Computer Science and Technology 22*(3), 387–396.

Koplowitz, J. and T. A. Brown (1981). On the relation of performance to editing in nearest neighbor rules. *Pattern Recognition 13*(3), 251–255.

Lai, X., L. Wang, H. Tang, and F. a. Witzmann (2011, oct). A novel alignment method and multiple filters for exclusion of unqualified peptides to enhance label-

free quantification using peptide intensity in LC-MS/MS. *Journal of proteome research 10*(10), 4799–812.

Liu, K., J. Zhang, J. Wang, L. Zhao, X. Peng, W. Jia, W. Ying, Y. Zhu, H. Xie, F. He, and X. Qian (2009). Relationship between sample loading amount and peptide identification and its effects on quantitative proteomics. *Analytical Chemistry 81*(4), 1307–1314.

Lucas, J. E., J. W. Thompson, L. G. Dubois, J. McCarthy, H. Tillmann, A. Thompson, N. Shire, R. Hendrickson, F. Dieguez, P. Goldman, K. Schwarz, K. Patel, J. McHutchison, and M. A. Moseley (2012). Metaprotein expression modeling for label-free quantitative proteomics. *BMC Bioinformatics 13*(1), 1.

Miranda, A. L. B., L. P. F. Garcia, A. C. P. L. F. Carvalho, and A. C. Lorena (2009). Use of classification algorithms in noise detection and elimination. In E. Corchado, X. Wu, E. Oja, Á. Herrero, and B. Baruque (Eds.), *Hybrid Artificial Intelligence Systems*, Berlin, Heidelberg, pp. 417–424. Springer Berlin Heidelberg.

Morales, P., J. Luengo, L. P. F. Garcia, A. C. Lorena, A. C. de Carvalho, and F. Herrera (2016). NoiseFiltersR: Label Noise Filters for Data Preprocessing in Classification.

Muhlenbach, F., S. Lallich, and D. A. Zighed (2004). Identifying and Handling Mislabelled Instances. *Journal of Intelligent Information Systems 22*(1), 89–109.

Pasquier, F., S. Delany, and P. Cunningham (2005). Blame-based noise reduction: An alternative perspective on noise reduction for lazy learning. In *Dublin, Trinity College Dublin, Department of Computer Science*, pp. 1–17.

Polpitiya, A. D., W. J. Qian, N. Jaitly, V. A. Petyuk, J. N. Adkins, D. G. Camp, G. A. Anderson, and R. D. Smith (2008). DAnTE: A statistical tool for quantitative analysis of -omics data. *Bioinformatics 24*(13), 1556–1558.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning 1*(1), 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

R Core Team (2017). R: A Language and Environment for Statistical Computing.

Sáez, J. A., M. Galar, J. Luengo, and F. Herrera (2014). Analyzing the presence of noise in multi-class problems: Alleviating its influence with the One-vs-One decomposition. *Knowledge and Information Systems 38*(1), 179–206.

Sánchez, J. S., F. Pla, and F. J. Ferri (1997). Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognition Letters 18*(6), 507–513.

Schader, M. and F. Schmid (1989). Two Rules of Thumb for the Approximation of the Binomial Distribution by the Normal Distribution. *The American Statistician 43*(1), 23–24.

Schapire, R. E. (1997). Using output codes to boost multiclass learning problems. In D. H. Fisher (Ed.), *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, Nashville, pp. 313–321. Morgan Kaufmann Publishers Inc.

Silva, J. C., M. V. Gorenstein, G. Z. Li, J. P. Vissers, and S. J. Geromanos (2006a). Absolute quantification of proteins by LCMSE: A virtue of parallel MS acquisition. *Molecular and Cellular Proteomics 5*(1), 144–156.

Silva, J. C., M. V. Gorenstein, G.-Z. Li, J. P. C. Vissers, and S. J. Geromanos (2006b, jan). Absolute quantification of proteins by LCMSE: a virtue of parallel MS acquisition. *Molecular & cellular proteomics : MCP 5*(1), 144–56.

Sluban, B., D. Gamberger, and N. Lavra (2010). Advances in class noise detection. *Frontiers in Artificial Intelligence and Applications 215*, 1105–1106.

Smith, L. M. and N. L. Kelleher (2013). Proteoform: a single term describing protein complexity Lloyd. *Nat Methods 10*(3), 186–187.

Smith, M. R. and T. Martinez (2011). Improving classification accuracy by identifying and removing instances that should be misclassified. In *Proceedings of the International Joint Conference on Neural Networks*, pp. 2690–2697.

Steen, H. and M. Mann (2004, sep). The ABC's (and XYZ's) of peptide sequencing. *Nature reviews. Molecular cell biology 5*(9), 699–711.

Suomi, T., G. L. Corthals, O. S. Nevalainen, and L. L. Elo (2015). Using peptide-level proteomics data for detecting differentially expressed proteins. *Journal of Proteome Research 14*(11), 4564–4570.

The UniProt Consortium (2017, jan). UniProt: the universal protein knowledgebase. *Nucleic Acids Research 45*(D1), D158–D169.

Tomek, I. (1976). An Experiment with the Nearest-Neighbor Rule. *IEEE Transactions on Systems, Man, and Cybernetics SMC-6*(6), 448–452.

Verbaeten, S. (2002). Identifying mislabeled training examples in ILP classification problems. In *Proceedings of twelfth Belgian-Dutch conference on machine learning*, pp. 71–78.

Verbaeten, S. and A. V. Assche (2003). Ensemble Methods for Noise Elimination in Classification Problems. *Proceedings of the 4th international conference on Multiple classifier systems (MCS'03) 2709*, 317–325.

Vitek, O. (2009, may). Getting started in computational mass spectrometry-based proteomics. *PLoS computational biology 5*(5), e1000366.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association 58*(301), 236–&.

Webb-Robertson, B.-J. M., M. M. Matzke, S. Datta, S. H. Payne, J. Kang, L. M. Bramer, C. D. Nicora, A. K. Shukla, T. O. Metz, K. D. Rodland, R. D. Smith, M. F. Tardiff, J. E. McDermott, J. G. Pounds, and K. M. Waters (2014). Bayesian Proteoform Modeling Improves Protein Quantification of Global Proteomic Measurements. *Molecular & Cellular Proteomics 13*(12), 3639–3646.

Wheway, V. (2001). Using Boosting to Detect Noisy Data. In *Revised Papers from the PRICAI 2000 Workshop Reader, Four Workshops Held at PRICAI 2000 on Advances in Artificial Intelligence*, Berlin, Heidelberg, pp. 123–132. Springer-Verlag.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEE Transactions on systems, man, and cybernetics 2*(3), 408–421.

Zhu, X. and X. Wu (2004). Class Noise vs. Attribute Noise: A Quantitative Study. *Artificial Intelligence Review 22*(3), 177–210.

Zhu, X., X. Wu, and Q. Chen (2003). Eliminating class noise in large datasets. In T. Fawcett and N. Mishra (Eds.), *Proceedings of the Twentieth International Conference on Machine Learning*, Washington D.C., pp. 920–927.

CURRICULUM VITAE

Melissa C. Key

EDUCATION

- Ph.D. in Biostatistics, Indiana University, Indianapolis, IN, 2020
  (minor in Life Science)
- M.S. in Statistics, Purdue University, West Lafayette, IN, 2005
- B.S. in Mathematics, University of Chicago, Chicago, IL, 2003
  (specialization in economics)

WORKING EXPERIENCE

- Research Assistant, Indiana University, Indianapolis, Indiana June 2010 – March 2019
- Research Assistant, Purdue University, West Lafayette, Indiana, June 2007 – June 2010
- Teaching Assistant, Purdue University, West Lafayette, Indiana, January 2004 – December 2009

HONORS, AWARDS AND FELLOWSHIPS

- Outstanding Beginning Graduate Student Award in Biostatistics, April 13, 2013

SELECT PUBLICATIONS

- Ragg, S., **Key, M.**, Rankin, F., and WuDunn D., The effect of molecular weight on passage of proteins through the blood-aqueous barrier. *Investigative ophthalmology and visual science* 2019 60(5), 1461-1469

- Ragg, S., **Key, M.**, Rankin, F., and Hulbert, M., Insights from comparative serum proteomic profiling of children with sickle cell disease: the effect of hydroxyurea and genotype on Protein abundance. *Blood* 2016 128(22), 1302 - 1302

- Ragg, S., **Key, M.**, Rankin, F., Heiny, M., and Hulbert, M., Serum protein abundance in children with sickle cell disease at baseline, during acute pain crisis, and on hydroxyurea - compared to children with other pediatric diseases. *Blood* 2014 124(21), 4050-4050

- **Key, M.**, A tutorial in displayng mass spectrometery-based proteomic data using heat maps. *BMC Bioinformatics* 2012, 13(Suppl 16):S10

- Clough, T., **Key, M.**, Ott I., Ragg S., Schadow G., and Vitek, O., Protein quantification in label-free LC-MS experiments. *Journal of Proteome Research* 2009 8(11), 5275-5284